# DEVELOPING AN APP FOR STREAMLINED INVENTORY TRACKING WITH BARCODE SCANNING AND LOAD PLANNING OPTIMIZATION

**TENG YAN XIN**

**UNIVERSITI TUNKU ABDUL RAHMAN**

# DEVELOPING AN APP FOR STREAMLINED INVENTORY TRACKING WITH BARCODE SCANNING AND LOAD PLANNING OPTIMIZATION

**TENG YAN XIN**

**A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Software Engineering (Honours)**

**Lee Kong Chian Faculty of Engineering and Science**
**Universiti Tunku Abdul Rahman**

**September 2025**

# DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Name : TENG YAN XIN

ID No. : 2106670

Date : 18 September 2025

## COPYRIGHT STATEMENT



This final year project report is submitted in partial fulfilment of the requirements for the degree of Software Engineering at Universiti Tunku Abdul Rahman (UTAR). This final year project report represents the work of the author, except where due acknowledgement has been made in the text. No part of this final year project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

# ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor, Ms. Beh Hooi Ching and my moderator, Dr Chia Kai Lin for their invaluable advice, guidance and enormous patience throughout the development of the project. Their continuous support and encouragement have been instrumental in ensuring the successful completion of this work.

I am also sincerely thankful to my peers and friends for their cooperation, assistance, and motivation during the course of this project. Their support has helped me overcome challenges and remain focused.

Lastly, I wish to extend my heartfelt appreciation to my family for their unconditional love, patience, and support. They have provided full support and encouraged me whenever I faced any difficulty during the project's development.

# ABSTRACT

Inventory management and load planning are important processes for organizations that handle large volumes of goods. However, many small and medium-sized enterprises (SMEs) still rely on manual record-keeping and random cargo loading practices due to the high cost and complexity of existing systems. These outdated practices often result in inaccurate stock records, inefficient use of vehicle space, and delays in distribution caused by time-consuming and unstructured load adjustments. To address these challenges, this project developed a Streamlined Inventory Tracking Application that integrates barcode scanning for fast and accurate stock management with an optimized load planning module. The application was implemented using React Native for mobile development and Firebase Firestore as the backend database to enable real-time data synchronization, while a binary tree bin packing algorithm was applied to generate efficient cargo loading arrangements. The methodology combined throwaway prototyping and incremental development, ensuring continuous refinement based on feedback and iterative improvements. The system was tested for functionality, usability, and performance, demonstrating improved stock accuracy, reduced manual workload, and improved space utilization compared to traditional manual methods. The results indicate that the proposed system is both affordable and practical for SMEs, offering a user-friendly solution that enhances operational efficiency. It is recommended that future improvements include focus on role-based access control, advanced reporting, and support for irregular cargo shapes to further increase usability and applicability.

Keywords: Inventory Management, Barcode Scanning, Load Planning, Binary Tree Bin Packing Algorithm, Mobile Application Development.

Subject Area: T58.5–58.64 Information Technology

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF SYMBOLS / ABBREVIATIONS

| | |
|---|---|
| SMEs | Small and Medium-Sized Enterprises |
| PDF | Portable Document Format |
| QR Code | Quick Response Code |
| ID | Identity |
| IT | Information Technology |
| SDLC | Software Development Life Cycle |
| ERP | enterprise resource planning |
| iOS | iPhone Operating System |
| CRUD | Create, Read, Update, Delete |
| 2D | Two dimensional |
| 3D | Three dimensional |
| $ | dollar sign |
| XML | Extensible Markup Language |
| € | The euro, EUR |
| API | Application programming interface |
| EMSs | Empty Maximal Spaces |
| JKR | Public Works Department |
| JPJ | Road Transport Department |
| GVW | Gross Vehicle Weight |
| BRKGA | Biased Random-Key Genetic Algorithm |
| GA | Genetic Algorithm |
| DBL | Deepest-Bottom-Left |
| WBS | Work Breakdown Structure |

| | |
|---|---|
| SDK | Software Development Kit |
| IDE | Integrated Development Environment |
| MySQL | My Structured Query Language |
| NoSQL | Not only SQL |
| EAN code | European Article Number code |

# LIST OF APPENDICES

# CHAPTER 1

# INTRODUCTION

## 1.1 General Introduction

Inventory management is a very important aspect within an organization that deals with large volumes of raw materials or physical goods, whether in manufacturing, retail, or distribution (Madamidola et al., 2024). The purpose of inventory management is to be easy and efficient for organization to manage the ordering, stocking, storing, and using of inventory. By managing inventory effectively, organizations are able to always know what items are in stock, how many items have, and where they are located.

Nowadays, there are many web-based inventory management systems available. However, many small and medium-sized enterprises (SMEs) still rely on manual record-keeping techniques, such as using Excel sheets to track inventory and plan cargo arrangement. This is primarily because today's load planners and inventory systems are expensive and complex, and therefore unacceptable to small and medium-sized businesses with tighter budgets and simpler operational requirements.

While these methods are commonly used, they do not provide real-time updates, requiring employees to physically walk through storage sites, verify stock levels, and manually update records on paper (Chan, Sathiapriya and Razali, 2023). This inefficient method may result in problems including miscounting stock or recording incorrect quantities, leading to inaccurate inventory records. Without real-time updates, inventory records may not reflect actual stock levels, leading to discrepancies between recorded and physical inventory (Barratt, Kull and Sodero, 2018).

Similarly, a lot of workers load cargo into vehicles using a random stacking technique, placing the items inside without following any structured plan. Their main goal is to fit everything inside the lorry, generally without considering factors such as load safety, weight distribution, or space optimization. Longer loading times, ineffective use of space, and potential damage to delicate items are all results of this disorganized approach.

To address these challenges, this project aims to develop a Streamlined Inventory Tracking Application with barcode scanning and load planning optimization. By providing real-time inventory updates, automating tracking processes, and optimizing load planning, the system will improve stock accuracy, reduce manual workload, and enhance overall efficiency in inventory and transportation management. The most important thing is to create software that is affordable for small and medium-sized enterprises.

## 1.2 Problem Statements

The literature review and the interview conducted revealed several problems associated with the currently available solutions for inventory tracking and load planning. The problems are inaccuracy of manual record-keeping, time-consuming and difficult to update load plan, and underutilized vehicle capacity.

### 1.2.1 Inaccuracy and Inefficiency of Manual Record-Keeping

One of the most common problems associated with manual inventory management is inaccurate record-keeping. This is because manual inventory tracking methods affect the visibility into actual goods in inventory and are prone to frequent discrepancies. Manual inventory tracking methods, reliant on physical inspections and paper-based records, making it difficult to maintain an accurate view of stock levels. Employees must physically visit storage locations to count inventory, a process prone to human error, such as miscounting or incorrect data entry. Employees may only visit storage sites weekly or biweekly to perform inventory counts, which can lead to overstocking or understocking issues going unnoticed and unacted upon, resulting in harder to identify discrepancies (Chopra, 2021). The goods are not replenished on time, and when employees discover it, it is often too late to take action.

### 1.2.2 Time-Consuming and Difficult Adjustments in Manual Cargo Load Planning

Many companies utilize Excel's drawing functions to create visual representations of each cargo items according to its size and arrange the items within a larger grid, representing the container or vehicle. This approach helps

them visualize the load plan more effectively, but it can still be time-consuming and prone to errors due to manual calculations and adjustments. This process is time-consuming, especially for large or complex loads, as it requires repeated manual drawing every cargo. Workers manually draw and adjust each cargo's based on a scaled size range using Excel, to fit them into the container grid. Since most cargo sizes vary, they need to estimate and adjust each item's placement manually. If the initial arrangement is not optimal, they must repeatedly adjust placements. This trial-and-error process to fit all items properly takes significant time, especially when making multiple adjustments to optimize space.

### 1.2.3 Underutilized Vehicle Capacity

Many companies load cargo onto vehicles without any structured method or planning. The Logistic workers often load items based on convenience, prioritizing goods that are readily accessible at the front of the truck rather than following a strategic plan to maximize space. Without a planned method, logistic workers may fail to optimize the vehicle's payload, leaving substantial portions of the container or truck empty. Workers load the cargo into vehicles without considering any calculations or following specific guidelines. They neither evaluate the weight, size or shape of the goods before loading. This lack of planning increases transportation costs, as more trips are required to deliver the same volume of goods, resulting in higher fuel and labor expenses. Furthermore, randomly placing cargo can lead to uneven weight distribution, creating unstable loads that risk shifting or overloading during transit. Such instability not only causes costly delays but also increases the likelihood of accidents, posing safety hazards on the road (Gaur, 2023).

## 1.3 Project Objectives

The main goal of this project is to solve the identified problems in Section 1.2 by developing a software solution. This solution simplifies inventory management by providing barcode scanning when inventory is in and out and providing plan load optimization features. The objectives of the project are as follows:

1. To conduct a thorough study of algorithms for generating optimal cargo load plans for vehicles.
2. To develop a functional mobile app for inventory tracking with integrated cargo load planning and optimization features for vehicle space utilization.
3. To evaluate the developed mobile app with Unit Test, System Usability Scale (SUS) and User Acceptance Testing (UAT).

## 1.4 Proposed Solution

To solve the problem in manual inventory tracking and unstructured cargo loading, an Android-based mobile application was decided to develop. This application integrates barcode scanning features to streamline inventory management and apply automatic load planning algorithms to optimize cargo arrangement.

For inventory tracking, when users want to issue raw materials to the production line, users find the barcode on the goods and scan it using the mobile application. After scanning, the application will retrieve and display the relevant item information from the database, such as product thumbnail, name, current stock quantity and stock status. The user can then update the quantity issued and submit after confirming that the information is correct. Once submitted, the database is updated in real-time, eliminating the need for manual records, thus reducing the risk of human error.

Users able to view the changes in stock levels without having to refresh the app. By having the application, users can monitor the inventory levels and respond quickly to any issues at any time, and from anywhere. They can just open the application and take immediate action with a few clicks without having to go to the warehouse. Users also don’t have to rely on hundreds of paper sheets or manual records, as the application provides a centralized inventory system that includes all inventory data.

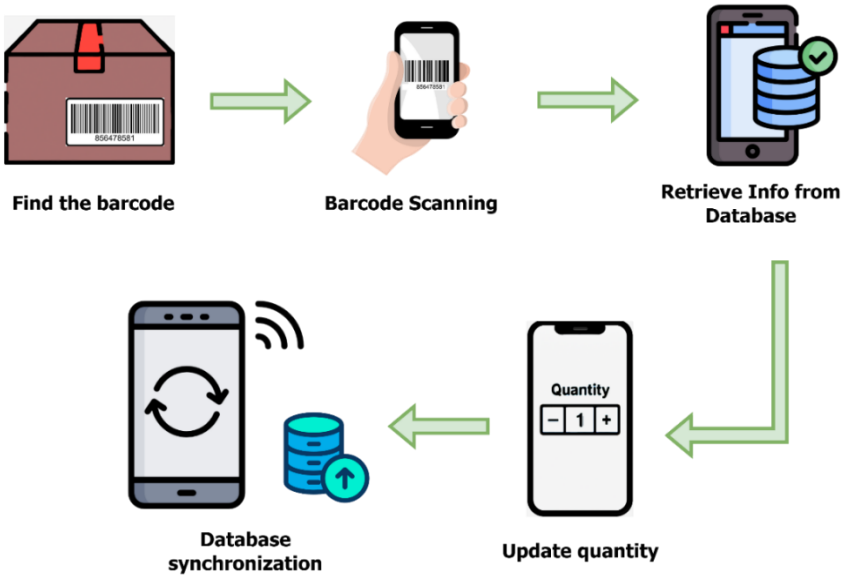



Figure1. 1: Operation flow of Barcode Scanning for Inventory Tracking System.

Besides, in terms of cargo load planning module, the application includes an automated feature that optimizes the arrangement of cargo during the loading process. By using this application, users no longer have to do manual calculations and can simply input the size of the container or lorry, as well as the dimension and quantity of the cargo to be arranged. Then the application will generate the cargo representations of corresponding proportional size and arrange them neatly in the lorry. By applying the packing algorithm, the lorry space can be maximally utilized.

User do not need to plan or draw scaled representations of each cargo one by one, which not only reducing labour costs and time, but also provides workers with a clear and structured loading guide. If the user needs to update the size of a previously provided cargo, they can directly edit the dimensions in the application, and such an operation does not take more than 1 minute. Then the system will recalculate the arrangement and update the load plan based on the updated information. Therefore, users are no longer to redraw or recreate the cargo to scale. Besides, users also can retrieve a previously generated load plan easily by scanning the QR code on the printed PDF document. This is because the barcode contains a unique load plan ID, which allows the application to fetch and show the corresponding load plan immediately.

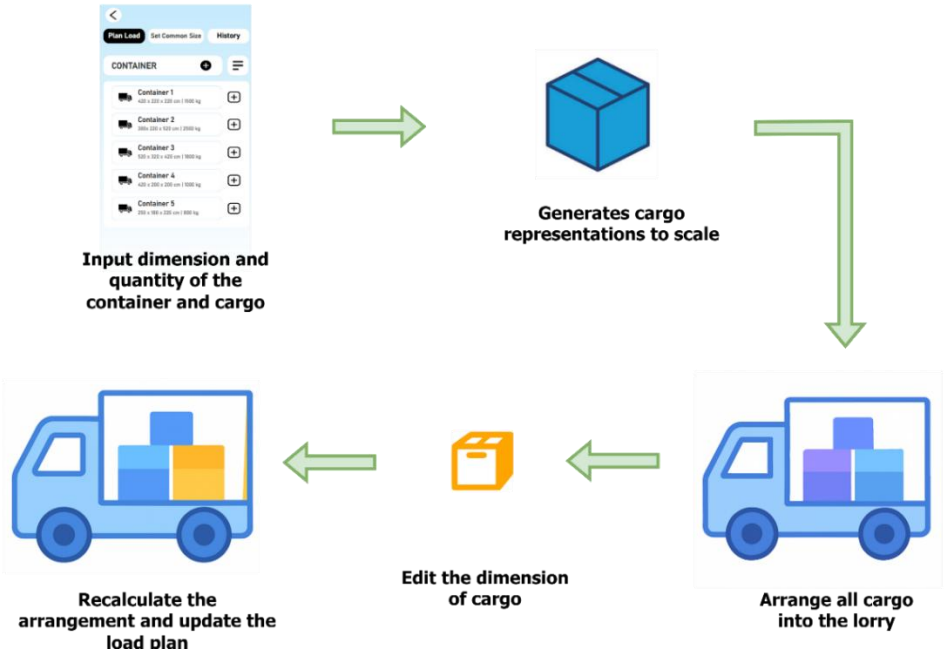



Figure1. 2: Operation Flow of Load Planning.

## 1.5 Proposed Approach

The combination of throwaway prototyping and incremental development methodologies was chosen to implement in this proposed system. Since the application's feature set is primarily based on personal ideas and the analysis of the current market situation, supplemented by a few user requirements collected through interviews. Therefore, in the absence of specific and complete user requirements, early developing an initial prototype for determining the core features and incrementally building feature by feature provide flexibility to changes in design and feature development.

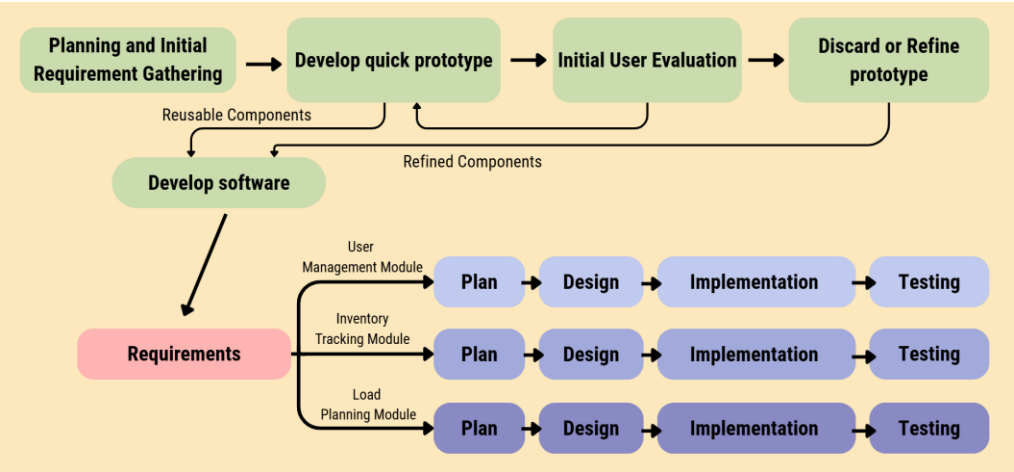



Figure1. 3: Overview of the Software Development Methodology.

Throwaway prototyping, also known as rapid prototyping, is used in the early stages of software development. A prototype of the final product is created quickly before actual software development takes place and then shared with users for feedback (Perera et al., 2022). This model was chosen because it is well suited for dealing with uncertain requirements, especially when the user does not know the exact project requirements in advance and the initial requirements are unclear. The software features defined in the early stages are based on personal ideas and investigation. So throwaway prototypes quickly create tangible representations of application functionality early in the development process, encouraging experimentation and trial and error, instead of being tethered to the first solution or design that comes along, making it easier to explore multiple ideas without commitment (Douglas, 2025). As a result, stakeholders can better visualize and interact with the prototype during the interview session to provide more accurate requirements, helping clarify expectations and provide effective feedback.

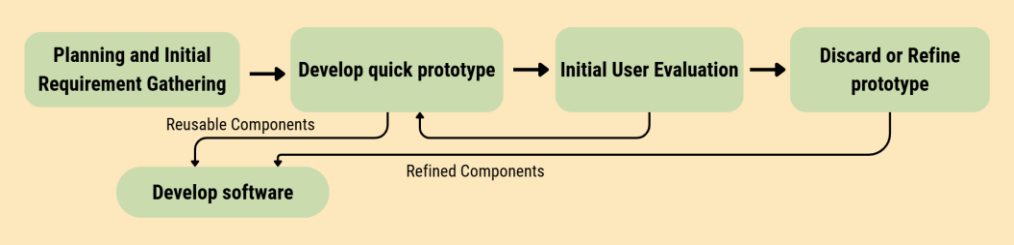



Figure1. 4: Overview of Throwaway Prototyping.

Besides, the end-users mostly do not have an IT background, therefore prototype can serve as a common visual language for diverse stakeholders. This reduces the risk of misunderstanding or misinterpreting of requirements, because users can intuitively see what the project is doing and what functions it currently has, regardless of technical background, so as to better propose their own ideas and define the user requirements. In conclusion, throwaway prototyping means building initial ideas for different applications, interfaces, or functions, without necessarily intending to include them in the final system. Instead, its purpose is to collect and gather feedback, determine the final functionality of the software, and prove that the concept can be implemented. Once the prototype has served its purpose, such as finalizing all

software functionality and design, it is set aside and then application development begins.

After analysing the feedback, the project transitioned from the prototyping phase to a more structured development approach. The feedback gained from early user testing, interviews, and prototype refinement helped shape a clearer understanding of user expectations and requirements. With these findings, the formal software development phase began, guided by a structured Software Development Life Cycle (SDLC). Each increment in the development process follows the SDLC phases including Planning, Design, Implementation, and Testing (GeeksforGeeks, 2025). The development is divided into three increments, each focusing on different core functionalities. The first increment focused on implementing the User Management Module, with the main objective of developing a secure registration and login system for user authentication. The second increment focused on developing the Inventory Tracking Module, which allows users to manage stock efficiently through barcode scanning and record inventory in and out activities with real-time updates. Lastly, the third increment focused on the development of the Load Planning Module, which allows the users to optimize the arrangement of cargo within transportation containers or lorry.

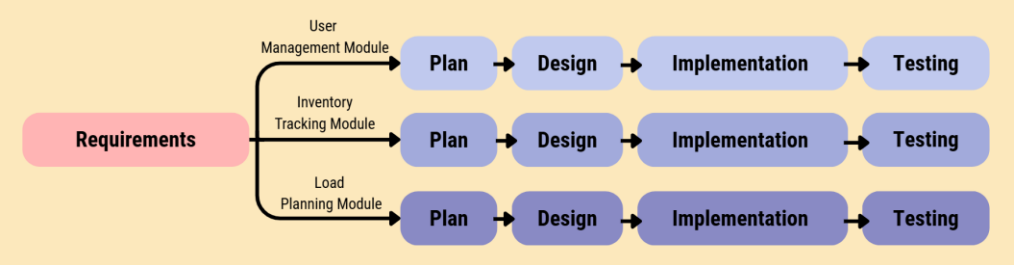



Figure1. 5: Overview of Incremental Process Model.

## 1.6 Scope and Limitation of the Project

### 1.6.1 Target End-Users

The project's target end users include storekeepers, warehouse staff, logistics personnel, and companies across various sectors such as retail, manufacturing, and wholesale, all of which require efficient solutions for inventory management and cargo load planning to optimize their operations. The

application allows storekeepers and warehouse employees to track goods coming in and out of the warehouse by scanning barcodes on the items using the mobile application, monitoring stock levels and take immediate action anytime, anywhere.

Besides, Logistics personnel can use the application to plan the arrangement of goods within containers. With the cargo load planning functionality, logistics personnel can optimize the load planning process by automatically generating optimal layouts for cargo based on the size and dimensions of the goods. This ensures that the space within vehicles or containers is utilized as efficiently as possible.

This project is tailored for small and medium-sized enterprises (SMEs) that operate on a smaller scale and have limited resources and budgets. These SMEs are often unable to fully utilize the full functionality of existing systems on the market. Therefore, these companies are often unwilling to invest too many resources and money in building or adopting such complex solutions that may be beyond their means. As a result, this software is intentionally designed with simplicity and efficiency in mind, offering only a few essential features that directly cater to specific needs.

### 1.6.2 Feature Scope and Exclusions

The application is designed with a simplified set of core features, such as barcode scanning for real-time inventory tracking and automated cargo load planning. To ensure ease of use and accessibility for small and medium-sized enterprises (SMEs), it intentionally excludes unnecessary and cumbersome features, therefore it may lack advanced functionalities found in more comprehensive inventory and logistics systems, such as real-time multi-warehouse synchronization, predictive analytics, or integration with external enterprise resource planning (ERP) platforms.

### 1.6.3 Assumptions and Constraints of Cargo

The application is developed based on several simplifying assumptions to maintain usability and efficiency for its target users. The assumption is that all cargo items are rectangular or square in shape and do not account for irregularly shaped objects, cylindrical items, or cargo with unconventional

forms. Additionally, since the app assumes all cargo has standard shapes, there is a possibility of wasted space if the actual items vary in shape and cannot fit perfectly together.

Additionally, the system does not consider other critical cargo management factors such as item fragility, weight distribution, or stacking limitations, which could be essential for more complex logistics scenarios.

### 1.6.4 Target Platform

Besides, the application is currently only available for Android devices, which limits the use of users of iOS or other platforms. Lastly, the system requires an internet connection for synchronization with the Firebase database. In offline environments or regions with poor connectivity, real-time data updates may be delayed or unavailable.

### 1.6.5 Modules

#### 1.6.5.1 User Management Module

The User Management Module is responsible for handling user access, account creation, and authentication within the system. This module ensures that only registered users can log in and interact with the application. During registration, an existing user can create new accounts for additional staff by assigning login credentials such as email and password. The Login feature then allows newly registered users to securely access the system using the credentials provided to them. Once logged in, all users are granted the same level of access to the system's features, including managing inventory and performing barcode scanning activities.

#### 1.6.5.2 Inventory Tracking via Barcode Scanning Module

The Inventory Tracking via Barcode Scanning module allows users to efficiently manage and monitor inventory items using barcode technology. This module enhances accuracy and efficiency in stock control by reducing manual entry and enabling real-time updates. For example, users can view a complete list of inventory items. They can filter items by category, stock status, or search using keywords across item attributes. They can also manually add new inventory items, edit or delete existing items. The users also can quickly

identify and update stock quantities by scanning items in or out with the device camera. If a scanned barcode doesn't match any existing item, the system will alert the user to prevent errors. In addition, the system automatically notifies users with a red indicator when stock levels fall below a predefined threshold, helping to prevent shortages and ensuring timely replenishment.

#### 1.6.5.3 Load Planning Module

The Load Planning Module is designed to generate a load plan that help users efficiently organize and manage the arrangement of cargo items into containers for transport. It ensures optimal space usage and reduces loading errors. This module allows users to input cargo dimensions and container sizes, after which the system automatically generates an optimized load plan using a binary tree bin packing algorithm. If predefined sizes are insufficient, user can manually enter custom dimensions for containers or cargo. This module also enables users to adjust and refine load plans by dragging and dropping cargo items within the container layout for better optimization. Each cargo item can be labelled with both a colour code and a text label for easy identification in the load plan layout, helping users clearly recognize each item during the loading process. Once the load plan is finalized, users can generate a printable PDF report that includes cargo details and a QR code linked to the load plan history. By scanning the QR code, users can quickly retrieve the full load plan details. Additionally, the system provides a checklist for staff to double-check the arrangement of each cargo item. Staff can mark them as 'arranged' after confirming they are loaded correctly.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1 Introduction

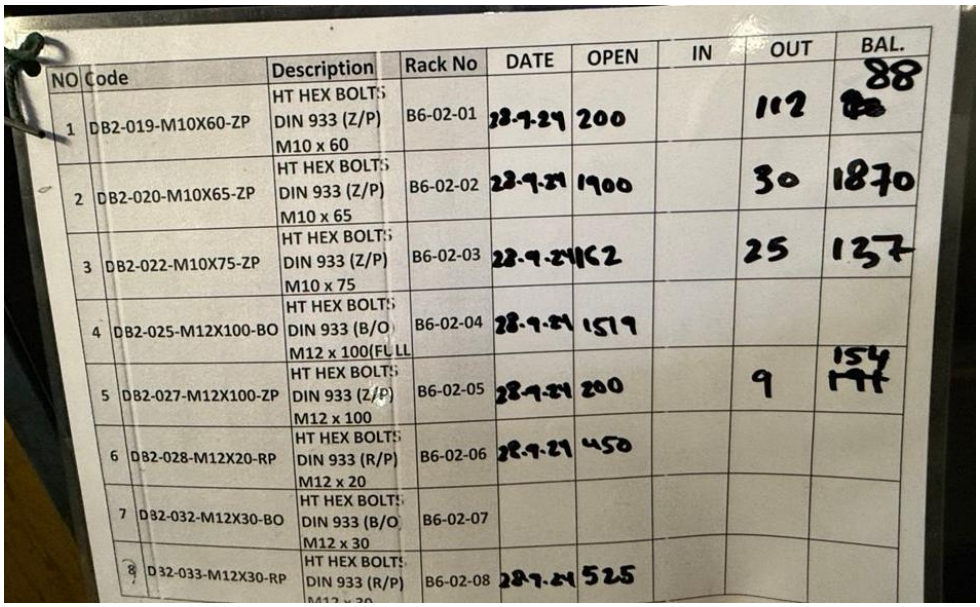
This chapter begins with an overview of traditional manual inventory tracking and load planning methods. It then reviews existing load planning applications to identify common features and functionalities relevant to this project's mobile application. Additionally, this chapter examines the algorithms commonly used in load optimization to guide the development of the proposed mobile application.

## 2.2 Overview of Traditional Manual Inventory Tracking Methods

Nowadays, many small and medium-sized enterprises (SMEs) still rely on manual inventory tracking methods, such as handwritten records, spreadsheets or stock cards to monitor and manage their stock. These approaches remain common primarily because they are less costly than implementing automated inventory management systems. Besides, these methods only require low initial investment and few technical resources, making them appeal to companies with tight budgets and limited access to technology. Despite the higher risk of mistakes, many SMEs expect to maintain accurate inventory counts and fast access to the analysis of inventory data without technological support.

The manual tracking process typically begins with physical stock counting. Warehouse staff must go to the storage areas and count each item manually. For example, when the staff want to know the latest inventory status, they have to go to the site to then find the exact location where the goods are placed, take out the goods and count them one by one (Setrag Shahikian, 2024). Finally, the quantities obtained during the physical count are manually recorded in logbooks, pre-printed forms, or spreadsheets such as Microsoft Excel (Kuhn, 2021). Spreadsheets may offer a small advantage over handwritten records by allowing for basic calculations and summaries through built-in formulas. However, they still rely entirely on manual data entry.

In addition, some companies use stock cards to monitor inventory movement. Each item has its own card, and staff members manually update the card every time stock is received, issued, or adjusted. To use a stock card effectively, staff must follow a structured process. When stock is received, the worker records the date, quantity received, and any relevant details on the card, then calculates the new balance by adding the received quantity to the previous balance. When stock is issued, such as when items are taken out for production or use, the worker notes the date, quantity issued, and purpose, then subtracts this amount from the previous balance to update the current stock level. This process ensures the stock card reflects the item's current inventory status. However, the system's accuracy hinges on consistent and immediate updates, which is a significant challenge. For instance, workers may take items out of storage without recording the transaction on the stock card, either due to time constraints, or lack of adherence to procedures. This failure to update records in real-time leads to discrepancies between the physical stock and the quantities recorded on the stock cards.



| NO | Code | Description | Rack No | DATE | OPEN | IN | OUT | BAL. |
|---|---|---|---|---|---|---|---|---|
| 1 | DB2-019-M10X60-ZP | HT HEX BOLTS DIN 933 (Z/P) M10 x 60 | B6-02-01 | 28-9-24 | 200 | | 112 | 88 |
| 2 | DB2-020-M10X65-ZP | HT HEX BOLTS DIN 933 (Z/P) M10 x 65 | B6-02-02 | 28-9-24 | 1900 | | 30 | 1870 |
| 3 | DB2-022-M10X75-ZP | HT HEX BOLTS DIN 933 (Z/P) M10 x 75 | B6-02-03 | 28-9-24 | 162 | | 25 | 137 |
| 4 | DB2-025-M12X100-BO | HT HEX BOLTS DIN 933 (B/O) M12 x 100(FULL | B6-02-04 | 28-9-24 | 1519 | | | |
| 5 | DB2-027-M12X100-ZP | HT HEX BOLTS DIN 933 (Z/P) M12 x 100 | B6-02-05 | 28-9-24 | 200 | | 9 | 154 191 |
| 6 | DB2-028-M12X20-RP | HT HEX BOLTS DIN 933 (R/P) M12 x 20 | B6-02-06 | 28-9-24 | 450 | | | |
| 7 | DB2-032-M12X30-BO | HT HEX BOLTS DIN 933 (B/O) M12 x 30 | B6-02-07 | | | | | |
| 8 | DB2-033-M12X30-RP | HT HEX BOLTS DIN 933 (R/P) | B6-02-08 | 28-9-24 | 525 | | | |

Figure 2. 1: Example of Stock Card

Besides, in order to speed up the inventory counting process, many staff may be assigned to perform inventory counts at the same time, which causes many employees to be unable to handle other tasks. Imagine if the company has tens of thousands of different goods, each with at least 1,000

quantities, manual tracking will be very time-consuming and laborious. Overall, manual inventory tracking involves a series of straightforward but labour-intensive tasks. Although it can be effective for small-scale operations, it becomes increasingly difficult to manage as inventory volume and complexity grow.

## 2.3 Manual Load Planning Techniques

Nowadays, many companies load cargo onto vehicles without applying any structured method or planning. Typically, logistical workers prioritize convenience over considerations such as maximizing space, balancing weight, or preventing damage. Small and medium-sized enterprises (SMEs) often relied on experience-based judgment, basic physical sketches, or simple spreadsheet tools to arrange cargo within containers or lorries. Items are also loaded based on their immediate availability rather than a systematic arrangement designed to optimize the load.

One of the most commonly used manual techniques is random stacking, where items are loaded into containers without a defined structure or layout. In such cases, the main goal is to fit everything inside the truck, rather than space efficiency or load safety. For instance, cargo may be loaded starting from the front of the container, and workers continue stacking until the space is filled. This approach can lead to uneven weight distribution, and inefficient use of available space.

Another common manual approach involves using tools like Microsoft Excel or grid paper to draw and simulate container layouts. According to the interviewee, workers start by using Excel's shape and drawing features to create basic visual blocks that represent each item of cargo, sized roughly according to their actual dimensions. These blocks are then placed onto a larger grid that represents the truck or container space. The goal is to visually arrange the cargo in a way that fits everything within the available area. Since cargo items often come in different shapes and sizes, the workers need to carefully estimate where each item might fit best. This usually means dragging and repositioning the blocks on the grid, one by one, trying different combinations to make everything work. If something doesn't fit or the layout seems off, the workers go back and move items around until a better

configuration is found. An example of the final cargo layout is shown in Figure 2.2.

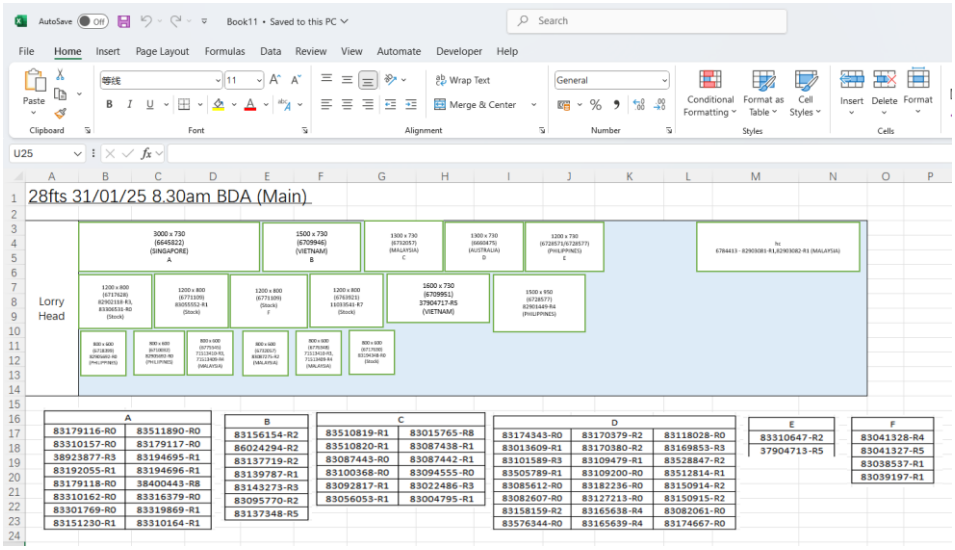



Figure 2. 2: Manual Sketch of Cargo Layout in Excel

## 2.4 Review of Existing Load Planner Applications

The cargo load planning applications such as GoodLoading, EasyCargo and CubeMaster are reviewed and analysed to identify the common features, strengths, and limitations of each application.

### 2.4.1 Common Features of Existing Load Planner Application

#### 2.4.1.1 Predefined Container Types and Cargo Sizes

Across the three load planning applications reviewed, several common features were identified. These applications offer a set of predefined commonly used container types and cargo sizes. Users can directly select from the predefined cargo items to put them into the container rather than manually inputting each item's details. They only need to adjust the quantity to generate a load plan, which is simple and convenient. Additionally, these applications offer flexibility by allowing users to define custom container sizes, including trailers, standard containers, air freight containers, and pallets. If a container or cargo size is not in the predefined list, users can also input custom dimensions.

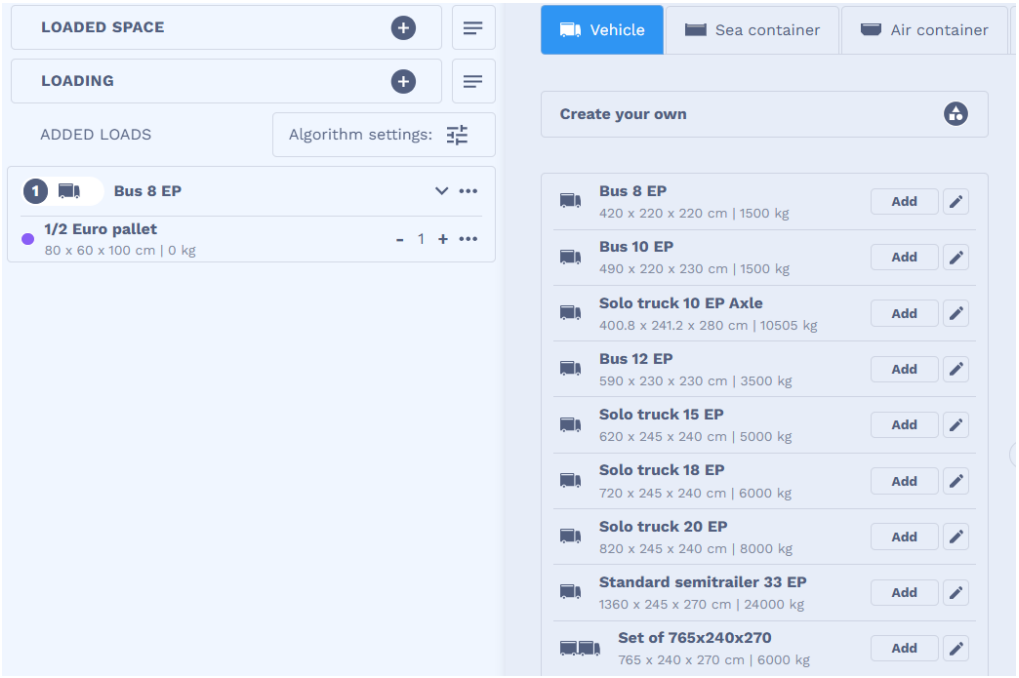



Figure 2. 3: Predefined Vehicle Selection of GoodLoading.

### 2.4.1.2 Space Calculation

Besides, three existing applications also have space calculation features. When a cargo is added to a container, the system automatically calculates both the occupied and remaining free space. This feature is able to ensure optimal space utilization by providing users with a clear overview of whether or not more cargo can be added.

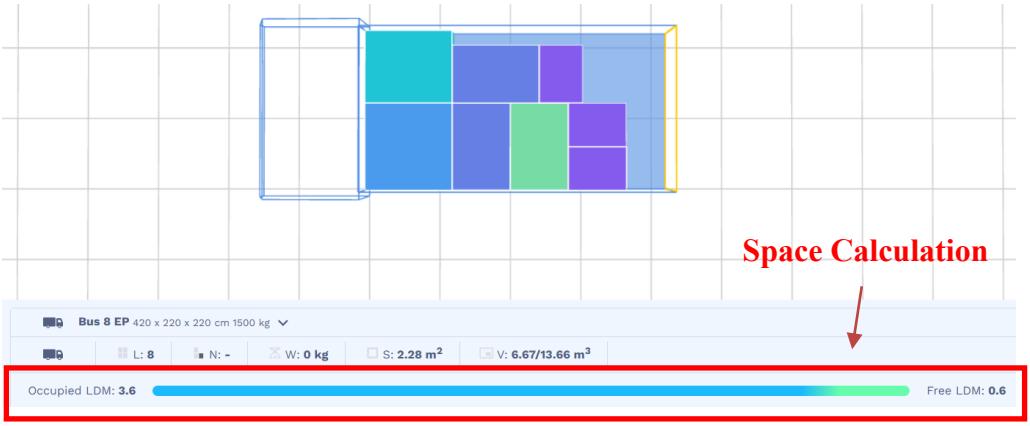



Figure 2. 4: Space Calculation Features of GoodLoading.


Figure 2. 5: Space Calculation Features of EasyCargo.

#### 2.4.1.3 Provide 3D Visualization

Furthermore, another common feature is providing 3D visualization load plan, which allows users to interactively view and adjust their load plans in three dimensions. It gives them the flexibility to zoom in, rotate, and adjust their view, providing a more immersive and intuitive way to let users get a better look at how the cargo is arranged inside the container. Rather than just looking at a 2D layout, users can switch between different views like top, front, or side to understand how everything fits together. Some apps also allow users to partially rotate the view, such as 180 or 360 degrees, while others limit fixed-angle views.

#### 2.4.1.4 Automated and Manual Adjustment Options

Another useful feature found in many load planning applications is the combination of automated and manual adjustment options. The system usually generates a loading plan automatically using built-in algorithms, which saves time and helps optimize space. However, users aren't limited to the system's suggestions. They can still manually adjust or reposition items if needed. Most applications support simple drag and drop functionality, allowing users to click, drag, and release items to position them exactly where they want inside the container.

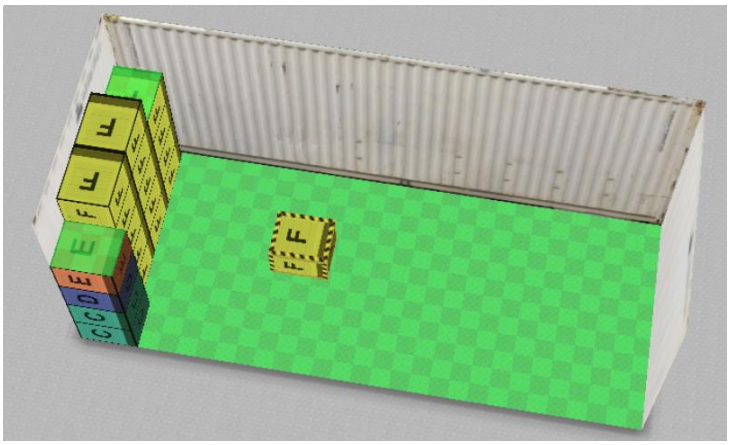


Figure 2. 6: Drag and Drop Features of EasyCargo.

#### 2.4.1.5 Export Options

Export options are a common feature among load planning applications, providing users with the ability to share, print, or analyze their load plans in different formats. The most commonly available options include PDF, Excel, Photo, and Sharable Link. PDF exports are standard for generating printable documents, a feature offered by all three applications. Excel exports are available only in EasyCargo, providing users with the ability to manipulate or analyze data in a spreadsheet. Photo exports offer a quick, visual way to share the load plan, and are available in GoodLoading. Sharable Links, which allow for easy real-time collaboration, are available in all three applications.

### 2.4.2 Advantages and Limitations of Existing Load Planner Applications

#### 2.4.2.1 GoodLoading

**Advantage:**

1. **Arrangement Algorithms:**

   A key advantage of GoodLoading over other load planner applications is its multiple algorithms for cargo arrangement. These features allow users to decide how to arrange the cargo in the container in order to generate a load plan. For example, the optimal arrangement algorithm is designed to maximize residual space, ensuring the most efficient use

of available container space, regardless of the order in which the cargo is added. Alternatively, GoodLoading offers an algorithm that arranges cargo in the order it was added, as well as one that prioritizes placing the heaviest items first. These diverse options allow users to tailor the load plan based on specific needs and preferences.

2. **Rotate The Container View 360 Degrees:**

GoodLoading allows users to rotate the container view 360 degrees by holding down the left mouse button, giving them the ability to inspect cargo arrangements from all angles for a more thorough review. In addition, GoodLoading features drag-and-drop functionality, enabling users to easily move cargo items and place them anywhere within the container.

**Limitations:**

1. **Limited Direct Rotation:**

One limitation of GoodLoading is the limited direct rotation functionality. Users cannot rotate the cargo directly within the displayed load plan using the mouse. Instead, to adjust the orientation of cargo, they must navigate to the cargo information panel and manually select the available options. This extra step can be a bit less intuitive compared to applications that allow direct, on-the-spot cargo rotation, potentially slowing down the user experience.

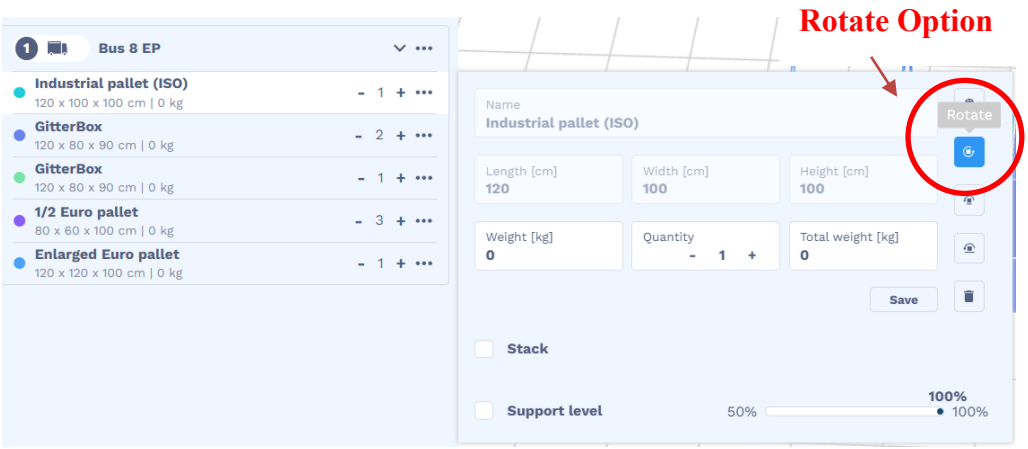



Figure 2. 7: Cargo Rotation via Cargo Information Panel.

2. **Lack of Step-by-Step Cargo Placement:**

Another limitation of GoodLoading is the lack of a step-by-step cargo placement process. When users add multiple cargo items to the load plan, the system automatically generates the entire loading arrangement in one action. There is no option to manually place each item one by one, nor is the placement process broken down into steps. As a result, users cannot track the exact sequence of placement in real-time. Instead, the system displays the placement sequence in a report, showing the order in which cargo was arranged. This means users must rely solely on the cargo names in the report to understand the sequence, making it less interactive and potentially confusing.

3. **Color-Coded Cargo Without Text Labels**

GoodLoading uses color-coding to differentiate between cargo types, and while it does provide text labels, they only appear when the user hovers the cursor over a specific cargo item. This means that at a glance, users can't immediately identify each cargo type without interacting with the interface.

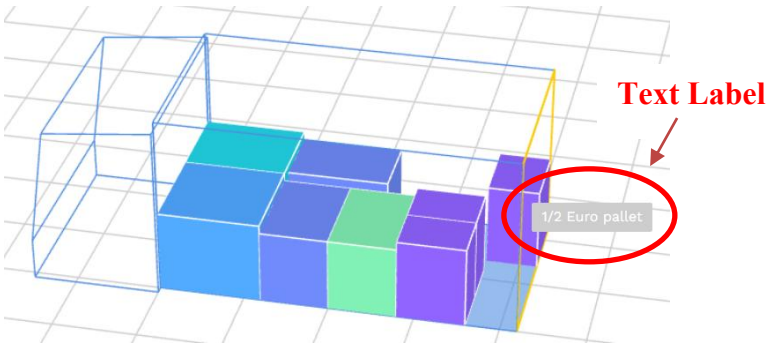



Figure 2. 8: Cargo Text Label Displayed on Hover in GoodLoading.

### 2.4.2.2 EasyCargo

**Advantages**

1. **Tutorial on Login:**

Upon logging in, EasyCargo offers a helpful tutorial that guides users through the process of generating a load plan. This feature is especially

useful for new users, as it helps them quickly understand the software's functions and get up to speed without feeling overwhelmed.

2. **Step-by-Step Placement Process:**

   In contrast to GoodLoading, EasyCargo offers a step-by-step cargo placement feature. This allows users to control the placement of cargo one group at a time, by clicking a button for each step, providing better flexibility and precision.

3. **Color-Coded with Visible Text Labels:**

   EasyCargo clearly labels cargo using both colors and visible text directly on the items, making it easy for users to identify cargo types at a glance without needing to hover or open additional panels.

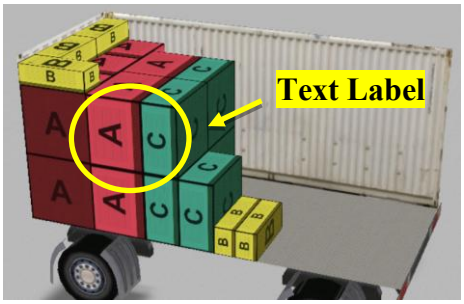



Figure 2. 9: Visible Text Labels and Color-Coded Cargo in EasyCargo.

**Limitations:**

1. **Limited Container View Rotation:**

   One drawback of EasyCargo is the limited container view rotation. Users can only rotate the container view up to 180 degrees, which may limit visibility and make it harder to precisely place cargo from different angles. This restriction can be frustrating for users who need a full 360-degree view to optimize their load plan.

#### 2.4.2.3 CubeMaster

**Advantages:**

1. **Multi-Language Support:**

   CubeMaster supports multiple languages, which makes the software accessible to users around the world. By offering various language options, CubeMaster ensures that international users can comfortably navigate the platform and generate load plans without language barriers, making it a versatile choice for global operations.

2. **Scenario Simulation:**

   CubeMaster offers a scenario simulation feature, allowing users to create and compare multiple loading scenarios. This tool helps users test different loading configurations and identify the most efficient plan. By simulating various arrangements, users can make data-driven decisions to optimize space and improve the overall load planning process.

**Limitations:**

1. **High Cost:**

   One significant downside of CubeMaster is its high cost, which may make it less accessible for smaller businesses or individual users. The most affordable subscription package starts at $49 per month, but it offers only limited features. Key functions such as Grouping, Palletizing, Balancing Rules, and Analysis View are unavailable in the standard package. Moreover, the standard package restricts important file management capabilities, such as uploading Excel and XML files or downloading reports in Excel and PDF formats. This can make it difficult for users to access the full range of tools unless they opt for a more expensive plan.

2. **Non-User-Friendly Interface:**

   Another drawback of CubeMaster is its non-user-friendly interface, which can make it difficult for new users to navigate and quickly learn how to use the software. Given the extensive range of features, the platform can

feel overwhelming, and users may require proper training to fully understand and utilize all its capabilities effectively.

3. **Limited Container and Cargo Rotation:**

   A limitation of CubeMaster is its restricted container and cargo rotation. Unlike other load planning software that allows free rotation, CubeMaster requires users to select specific views to see the container from different angles. Additionally, cargo cannot be directly rotated with a simple click-and-drag; instead, users must right-click and select a rotation option, making it less efficient.

4. **Lack of Clear Labeling:**

   Another limitation of CubeMaster is its lack of clear labeling. The software primarily uses color differentiation to distinguish between cargo types, which can be effective on screen but becomes problematic when printed in black and white. Without proper labels or text indicators, it becomes difficult to quickly identify and differentiate cargo types, especially in printouts. This can lead to confusion and errors during the load planning process, particularly when users rely on physical copies of the plan.

### 2.4.3 Key criteria for comparison

Table2. 1: Comparison of features between Existing Application.

| **Feature** | **GoodLoading** | **EasyCargo** | **CubeMaster** | **Proposed App** |
|---|---|---|---|---|
| **Cost** | €18/month, one user | $79/month, one user | $49/month, one user | Free |
| **User Interface** | Simple, but lacks intuitive navigation | User-friendly with login tutorials | Complicated for beginners | Userfriendly, simple for SMEs |
| **Predefined** | Yes | None | Yes | Yes |

| | | | | |
|---|---|---|---|---|
| **Container Types and Cargo Sizes** | | | | |
| **Step-by-step Cargo Placement** | None | Yes | Yes | Yes |
| **Drag and Drop Placement** | Yes | Yes | None | Yes |
| **Container Rotation** | 360° rotation | 180° rotation | Cannot be rotated; use selection to see each view | No direct rotation |
| **Cargo Rotation** | Through cargo information panel and use the available options | Rotate using directional controls | No direct cargo rotation | No direct cargo rotation |
| **Custom Cargo Size** | Yes | Yes | Yes | Yes |
| **Labeling & Color Coding** | Color-coded without text labels | Color-coded with text labels | Color-coded without text labels | Color-coded with text labels |
| **Export Options** | PDF, photo and sharable link | PDF, Excel file and sharable link | PDF and sharable link | PDF |
| **Language Support** | English only | English only | Multi-language support | English only |
| **Integration with Other Software** | API and offers tailored integration | API and SAP ERP | ERP/WMS integration | N/A |

| | | | | |
|---|---|---|---|---|
| | options | | | |
| **Accessibility** | Suitable for beginners or small-scale use | Suitable for medium-scale businesses | Best for large enterprises with complex logistics needs | Suitable for SMEs |

## 2.5 Rules and Constraints

### 2.5.1 Cargo Placement Constraints and Assumptions

1. Cargo must be placed firmly on the surface, suspension or oblique positioning is not allowed.
2. The length of each cargo item should not exceed the length of the container.
3. The effect of external force between cargo items is considered negligible.
4. Cargo is assumed to remain intact without deformation due to squeezing.
5. The cargo items are placed onto the container from largest to smallest based on volume.
6. The total weight of all cargo must not exceed the maximum load capacity of the container.

### 2.5.2 Cargo Orientations

In a 3D space, a rectangular or square cargo item can be placed in six possible orientations because of the six faces of a cuboid (BYJUS, n.d.). Each face can serve as a base, leading to six distinct placements.

Below is a diagram illustrating the six possible orientations for placing each cargo item:

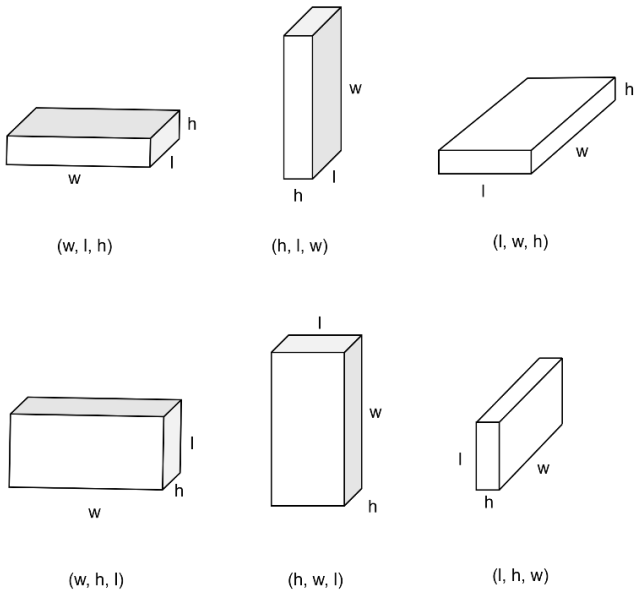



Figure 2. 10: Six cargo orientations.

### 2.5.3 Empty Maximal Space (EMS)

The maximal space is a way to represent the available empty space inside a bin., defined by their minimum and maximum coordinates. This only works if all objects are placed orthogonally. The rules for creating EMSs are volume check and dimension check. If the new EMS's volume is smaller than the smallest remaining box, it's discarded. If any dimension of the new EMS is smaller than the corresponding dimension of any remaining box, it's discarded. This strategy able to reduce the computational time by approximately 60%.

After placing the box in an EMS, the original EMS is split into six new EMS regions as shown in Figure 2.12 below that represent the remaining available space. The initial EMS is like a large container, and when the box is placed, the space to the left, right, above, below, in front, and behind the box can be seen as new available spaces (Saraiva, Nepomuceno and Pinheiro, 2015).

```
File  Edit  Format  Run  Options  Window  Help
#Use Maximal Space representation
#EMS = [
#    (x1, y1, z1), #minimum coodinate=bottom-left-back corner
#    (x2, y2, z2), #maximum coordinates= top-right-front corner

#minimum & maximum coordinates for intersected EMS
x1, y1, z1 = EMS[0]
x2, y2, z2 = EMS[1]

#minimum & maximum coordinates box1 to be placed
x3, y3, z3 = ems[0]
x4, y4, z4 = ems[1]

#When a box is placed inside an EMS,
#the remaining available space is
#divided into six new EMSs for 3D space
new_EMSs = [
    [(x1, y1, z1), (x3, y2, z2)], #Left EMS
    [(x4, y1, z1), (x2, y2, z2)], #Right EMS
    [(x1, y1, z1), (x2, y3, z2)], #Back EMS
    [(x1, y4, z1), (x2, y2, z2)], #Front EMS
    [(x1, y1, z1), (x2, y2, z3)], #Bottom EMS
    [(x1, y1, z4), (x2, y2, z2)]  #Top EMS
]
```

Figure 2. 11: Calculating Available Space Using EMS Representation.

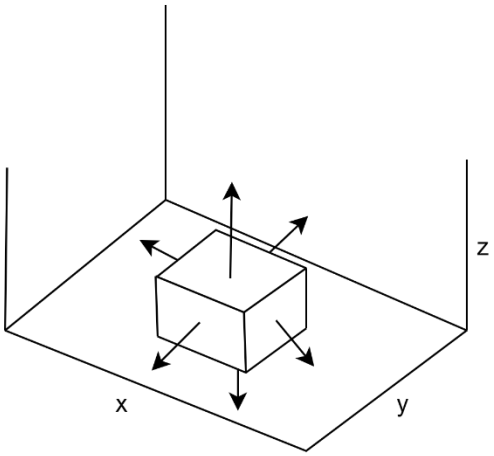



Figure 2. 12: Six EMS Directions After Loading a Cargo.

If a box placed at the corner of the intersected EMS, only three valid EMSs are generated due to the placement, which are the right, front, and top spaces remain. This is because the box touches the left, bottom, and back boundaries and occupies a full edge, face, or corner of the EMS.

```
#Use Maximal Space representation
#EMS = [
#    (x1, y1, z1), #minimum coodinate=bottom-left-back corner
#    (x2, y2, z2), #maximum coordinates= top-right-front corner

#minimum & maximum coordinates for intersected EMS
x1, y1, z1 = EMS[0]
x2, y2, z2 = EMS[1]

#minimum & maximum coordinates box1 to be placed
x3, y3, z3 = ems[0]
x4, y4, z4 = ems[1]

#three new EMSs for 3D space if ems[0] = EMS[0]
new_EMSs = [
    [(x4, y1, z1), (x2, y2, z2)], #Right EMS
    [(x1, y4, z1), (x2, y2, z2)], #Front EMS
    [(x1, y1, z4), (x2, y2, z2)]  #Top EMS
]
```

Figure 2. 13: Three New EMS when Box Placed at the Corner.

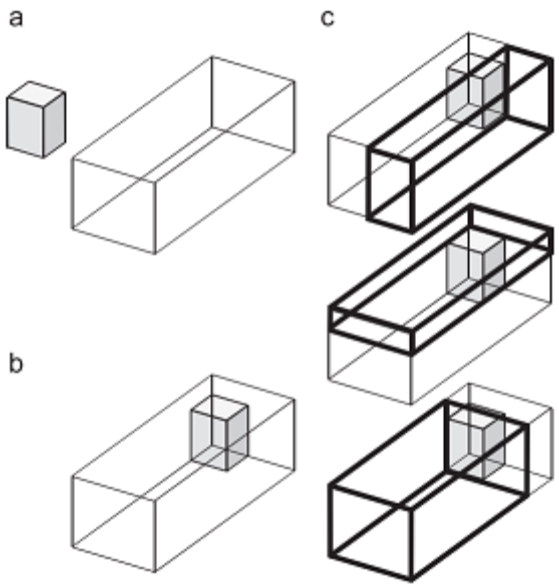



Figure 2. 14: New EMSs Resulting from the Placement of the Grey Box.

### 2.5.4 Load and Weight Restrictions for Vehicles

When planning the loading of cargo onto vehicles, it is critical to consider various load and weight restrictions to ensure safety, legal compliance, and operational efficiency. Since Malaysia enforces strict vehicle weight regulations, load planning must go beyond simply optimizing the physical arrangement of items within a container or vehicle. It must also ensure that the total weight of the cargo does not exceed the legal maximum allowed for the specific vehicle type.

According to Ministry of Transport Malaysia (2021), in Malaysia, these weight regulations are governed primarily under the Road Transport Act 1987 and enforced by the Road Transport Department (Jabatan Pengangkutan Jalan, JPJ) and the Public Works Department (JKR). The Gross Vehicle Weight (GVW) refers to the total weight of the vehicle, including its own weight, fuel, passengers, and cargo. Each vehicle type, depending on its axle configuration, has specific maximum allowable GVW limits (Asia-Pacific Economic Cooperation, 2017). The GVW limits is the maximum allowable weight the vehicle can carry. Below are the GVW limits for various vehicle types commonly used in Malaysia:

Table2. 2: Summary of GVW limits in Malaysia.

| Type of Vehicle | Number of Axle | Axle Configuration | GVW Limit (kg) |
|---|---|---|---|
| Rigid | 2 Axle (1 + 1) |  | 16,000 – 18,000 |
| | 3 Axle (1 + 2) |  | 20,000 – 25,000 |
| | 4 Axle (2 + 2) |  | 25,000 – 27,000 |
| Articulated | 3 Axle (1+1+1) |  | 26,000 – 30,000 |
| | 4 Axle (1+1+2) |  | 27,000 – 37,000 |
| | 5 Axle (1+1+3) |  | 27,000 – 39,000 |

## 2.6 Review of Load Planning Algorithms

### 2.6.1 Biased Random Key Genetic Algorithm

The Biased Random-Key Genetic Algorithm (BRKGA) is a variant of the genetic algorithm (GA) that combines the features of random-key

representation and biased selection from the population pool. In BRKGA, solutions are in the form of vectors of real numbers, also called random keys, between 0 and 1. These numbers are then translated into actual solutions, like a packing plan, using a problem-specific decoder. BRKGA has a biased selection process, it keeps the best solutions, called elite and combines them with others to create new solutions, while also adding some completely random solutions, which are mutants (Gonçalves and Resende, 2013). This balance helps BRKGA find great solutions without getting stuck.

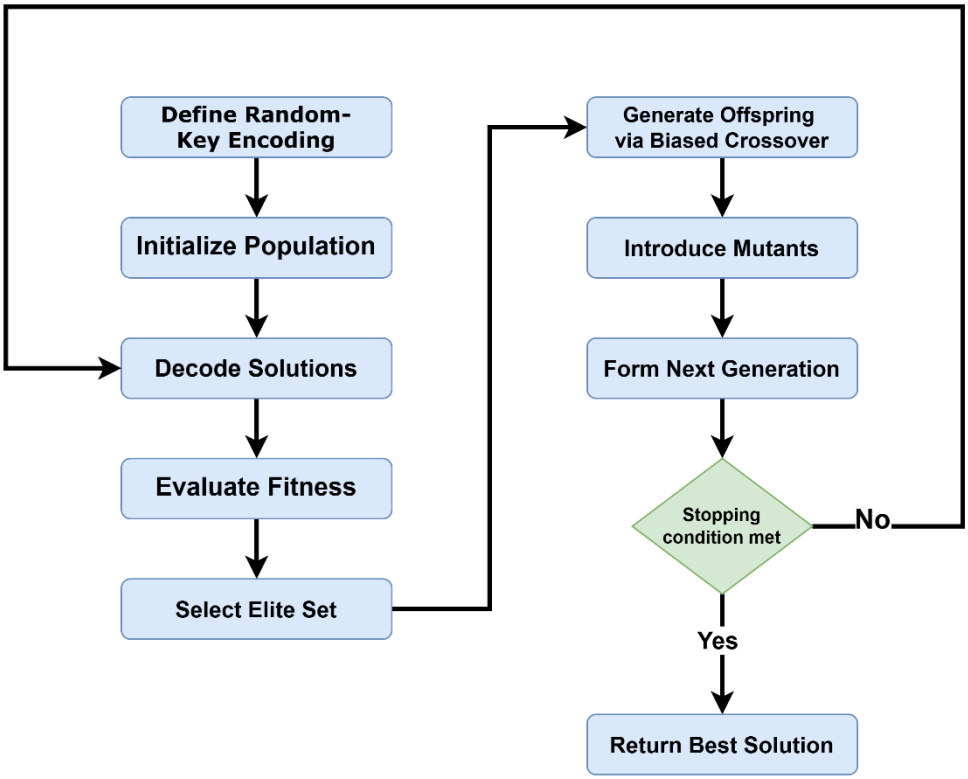



Figure 2. 15: Process of Biased Random-Key Genetic Algorithm.

The BRKGA follows a structured process to evolve a population of solutions toward an optimal or near-optimal state. The process starts by defining a random-key encoding, where each solution is a list of numbers between 0 and 1. These random keys correspond to features of the solution, such as item order or orientation in packing problems. Next, a set of solutions, called the population is randomly generated. Each individual is then decoded into a feasible solution using a problem-specific decoder and its quality is evaluated using a fitness function to see how good it is based on the problem's objective.

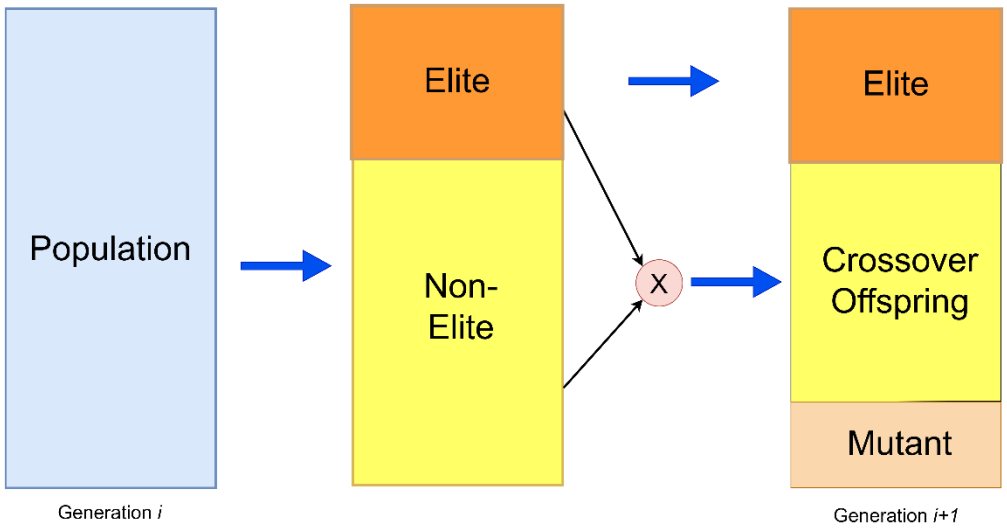



Figure 2. 16: Evolutionary process between consecutive generations.

Once all solutions are evaluated, they are ranked based on fitness, and a small subset of the best-performing individuals, known as the elite set, is preserved for the next generation. New solutions are generated using a biased crossover mechanism, where each offspring is formed by combining two parents (Londe et al., 2024). One from the elite set and one from the non-elite population. During crossover, there is a higher probability up to 70% that each gene in the offspring will be inherited from the elite parent, promoting the retention of high-quality traits. To maintain diversity and prevent premature convergence, a set of mutants which is completely random new individuals is also added to each new generation.

The next generation is formed by combining the elite individuals, the offspring from biased crossover, and the mutants, while keeping the population size constant. This process of decoding, evaluating, selecting elites, generating offspring, and introducing mutants is repeated across multiple generations. The algorithm continues until a predefined stopping criterion is met, such as a maximum number of generations, a time limit, or achieving a satisfactory solution. Ultimately, BRKGA returns the best solution found during its evolutionary search process.

With the BRKGA process clearly defined, the following section demonstrates its implementation using an example of 3D bin packing. There are 4 items need to be packed into a bin, focusing solely on the sequence in which items are placed. The bin dimensions are 10 x 10 x 10 cm, and the 4

items are: Item1 is 4 x 4 x 4; Item 2 is 6 x 4 x 3; Item 3 is 5 x 5 x 2 and Item 4 is 3 x 3 x 5.

Table2. 3: Dimension of Items to be Packed

| | Width (x) (cm) | Depth (y) (cm) | Height (z) (cm) |
|---|---|---|---|
| Item 1 | 4 | 4 | 4 |
| Item 2 | 6 | 4 | 3 |
| Item 3 | 5 | 5 | 2 |
| Item 4 | 3 | 3 | 5 |

In Step 1, we define the random-key encoding to represent a solution as a vector of 4 random keys, each a real number in [0, 1], corresponding to the packing order of the four items.

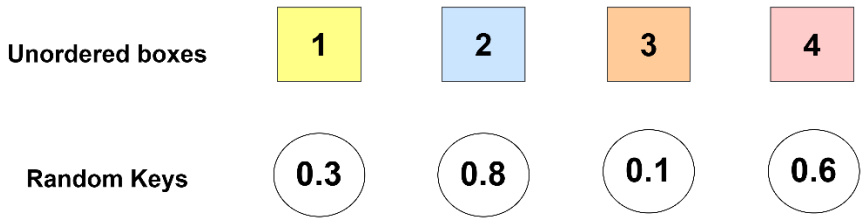



Figure 2. 17: Random Keys Generated.

In Step 2, a population of five individuals is initialized, each a 4-key vector, to create a diverse set of random solutions. With parameters specifying an elite set size of 1 (20%), a mutant set size of 1 (20%), and 3 offspring (60%), vectors such as [0.3, 0.8, 0.1, 0.6], [0.7, 0.2, 0.9, 0.4], [0.5, 0.1, 0.6, 0.8], [0.9, 0.4, 0.7, 0.2], and [0.2, 0.6, 0.3, 0.9] are generated.

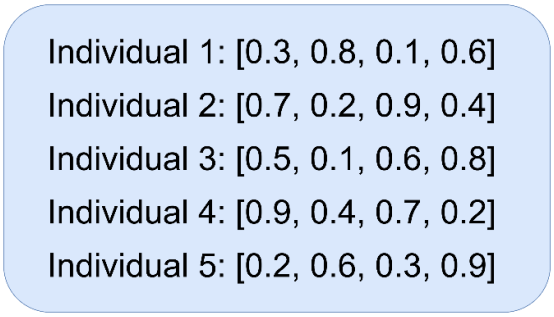



Figure 2. 18: Population of Five Individuals.

In Step 3, each solution is decoded into a packing configuration using a Deepest-Bottom-Left (DBL) heuristic. The 4 keys are sorted to determine the item sequence (e.g., for [0.3, 0.8, 0.1, 0.6], sorting [0.1, 0.3, 0.6, 0.8] results in indices [3, 1, 4, 2], meaning Item 3, Item 1, Item 4, Item 2).

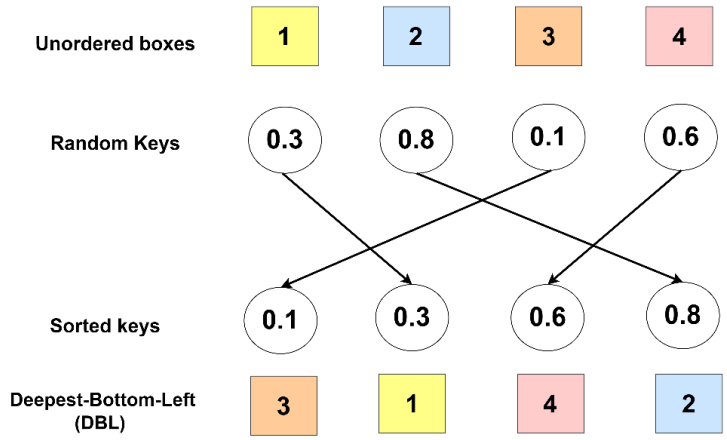



Figure 2. 19: Random Keys are Sorted.

The DBL heuristic places items in order into the first feasible bin, starting at (0,0,0) and moving to the deepest, bottommost, leftmost position, ensuring no overlap and no rotations. Item 3 (5×5×2) is placed at (0,0,0), and occupy from (0.0.0) to (5,5,2). Then Item 1 (4x4x4) is placed at (0,5,0) and occupy from (0,5,0) to (4,9,4). Next, Item 4 (3x3x5) placed at (5,0,0) and occupy from (5,0,0) to (8,3,5). Lastly, Item 2 (6x4x3) placed at (0,0,2) and occupy (0,0,2) to (6,4,5). This process is repeated for all individuals of the population to evaluate their packing solutions.

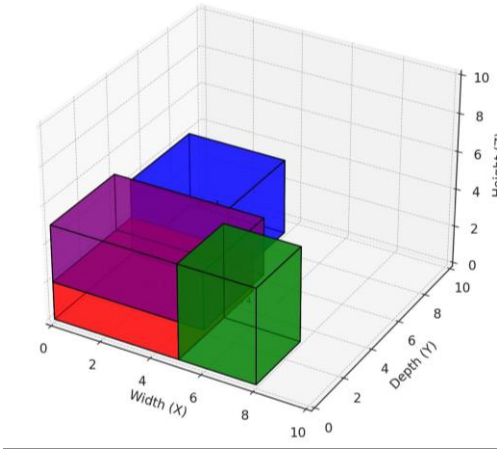



Figure 2. 20: Arrangement of First Individual.

Lastly, the fitness of the solution is evaluated to determine their quality based on the objective of maximizing the space utilization.

$$Utilization = \frac{Total\ Item\ Volume\ Packed}{Total\ Container\ Volume} \times 100\ \%$$

$$Total\ packed\ volume$$
$$= (5 \times 5 \times 2) + (4 \times 4 \times 4) + (3 \times 3 \times 5) + (6 \times 4 \times 3)$$
$$= 156$$

$$Container\ volume\ = 10 \times 10 \times 10 = 1000$$

$$Utilization = \frac{156}{1000} \times 100\ \% = 15.6\%$$

The utilization rate is 15.6%, it is not the optimal solution. After evaluating the fitness of all five individuals based on utilization, the individual with the best fitness is chosen as the elite solution and preserved in the next generation.

### 2.6.2 Binary Tree Bin Packing Algorithm

According to Gordon (2011), the Binary Tree Bin Packing Algorithm starts by placing the first (largest) block in the top left corner of the fixed rectangle, then split that rectangle into 2 smaller rectangles that represent the remaining spaces as right and below the placed block.

Each time a new block is placed, the remaining free space is further divided recursively. This process continues until all blocks are placed or no suitable space remains.

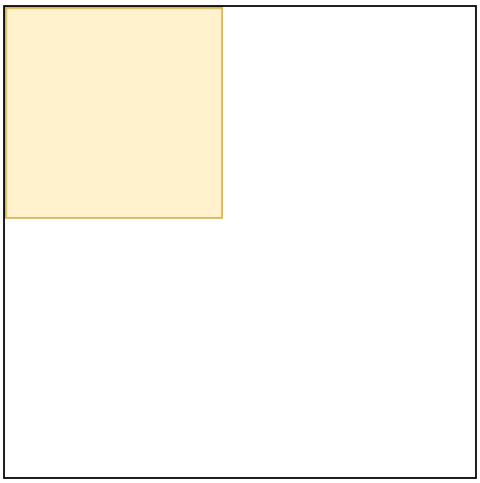


Figure 2. 21: Place the Largest Block in the Top Left Corner of the Rectangle.

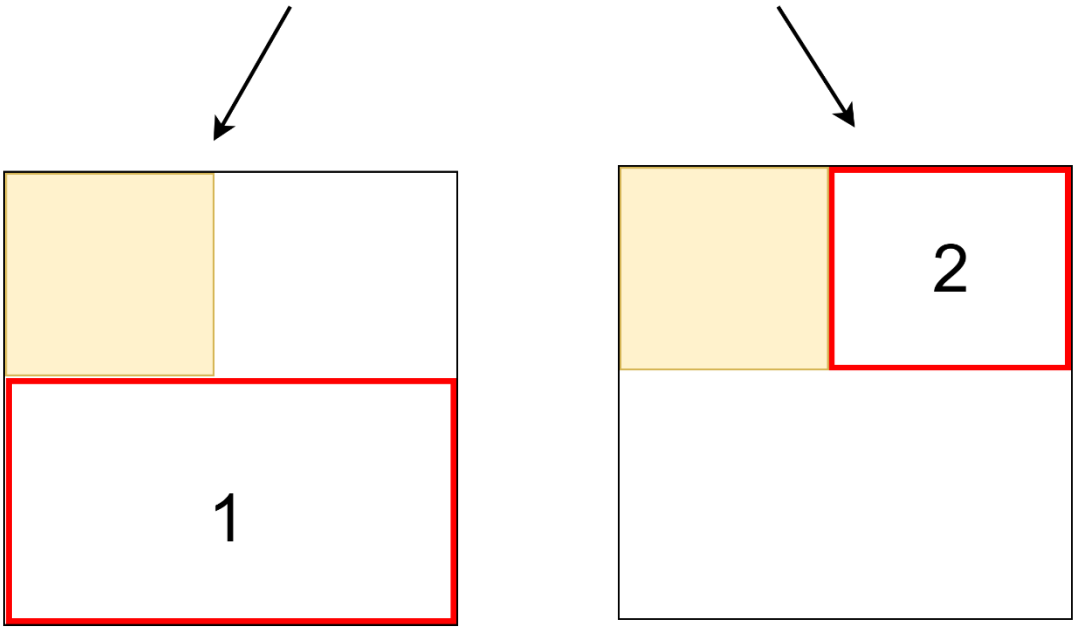



Figure 2. 22: Split Rectangle into 2 Smaller Rectangles.

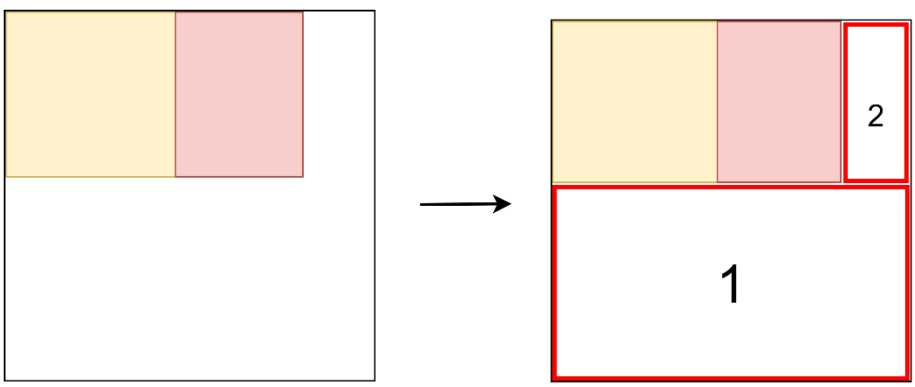



Figure 2. 23: Result of Placing the Second Largest Block.

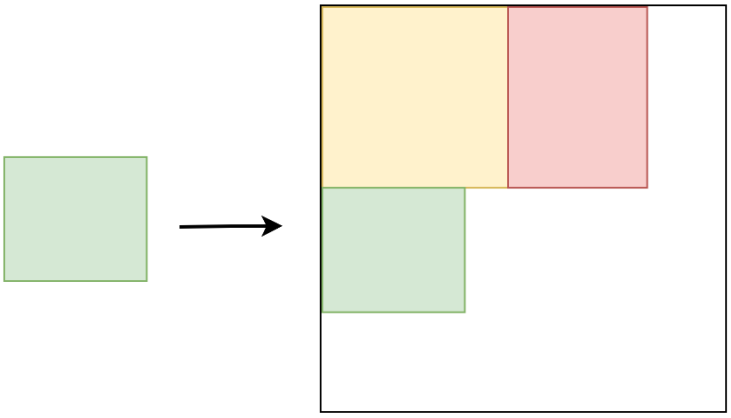


Figure 2. 24: Placing Third Block in the Bottom Remaining Space.

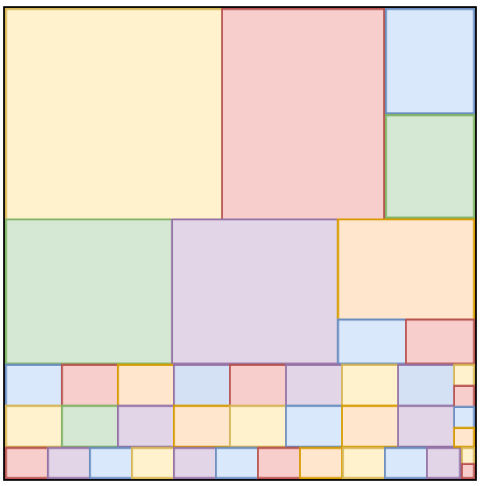


Figure 2. 25: Recursively Place All Blocks.

After understanding the concept of Binary Tree Bin Packing Algorithm, we can assume a binary tree can represent the container space as a hierarchy of subdivided sections. Each node represents a remaining space, and child nodes represent split sections where items are placed.

Steps for Binary Tree-Based Packing:

1. Start with a root node representing the full container space.
2. Place an item in the root node if it fits.
3. Split the remaining space into two child nodes:
   - One node represents the space next to the item (horizontal split).
   - The other represents the space below it (vertical split).

4. Repeat recursively until all items are packed or no more space is left.

Example:

Have a 100x100 container

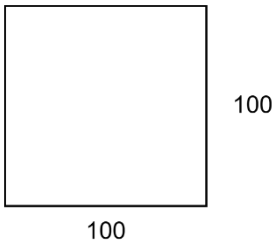



Figure 2. 26: 100x100 Container.

Different size items:

1. Item A (50x50) =2500
2. Item B (30x30) =900
3. Item C (20x70) =1400
4. Item D (10x10) =100

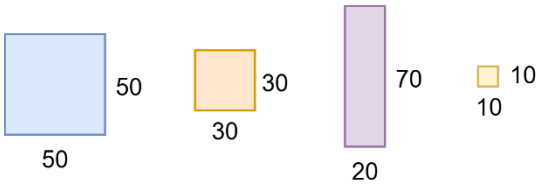



Figure 2. 27: 4 Items of Different Sizes.

1. Initial Container (100x100)

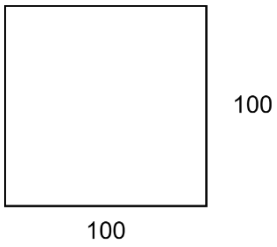



Figure 2. 28: 100x100 Container.

This is the root node of the binary tree, representing the entire available space.

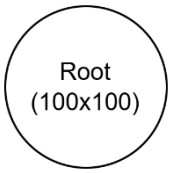



Figure 2. 29: Initial State of Binary Tree.

2. Place Largest Item (Item A (50x50))

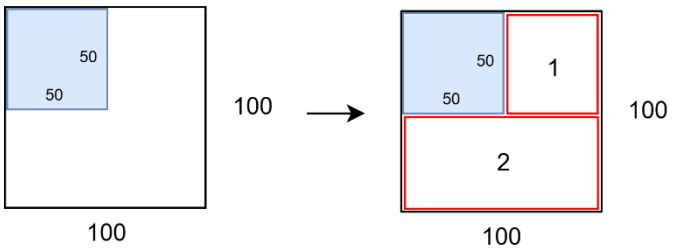



Figure 2. 30: Place Item A and the Remaining Spaces.

The first item A (50x50) is placed in the top-left corner. Container is split into two smaller rectangles:

- Right rectangle (Node): (50x50) empty
- Bottom rectangle (Node): (100x50) empty

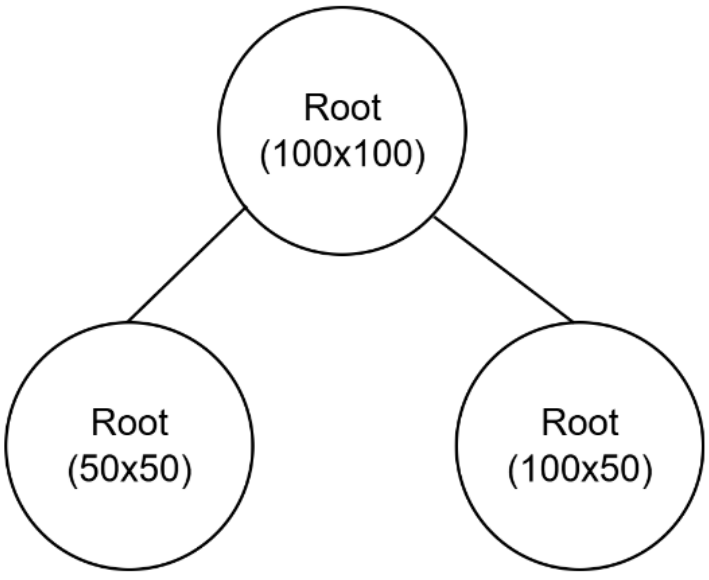


Figure 2. 31: Binary Tree.

3. Place Item C (20x70)

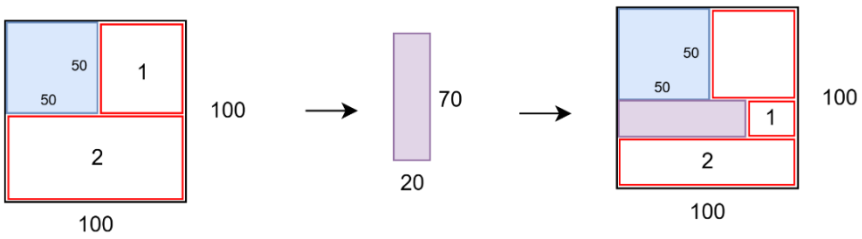



Figure 2. 32: Result of Placing Item C and the Remaining Spaces.

Item C (20x70) is placed in the remaining bottom space (100x50). Split it into new 2 rectangles:

- Right node: (30x20)
- Bottom node: (100x30)

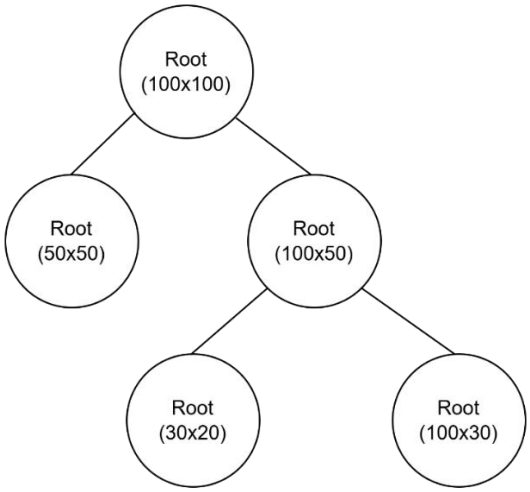



Figure 2. 33: Binary Tree.

4. Place Item B (30x30)

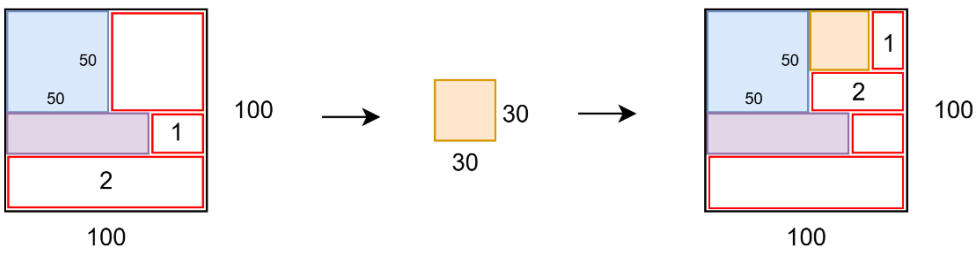



Figure 2. 34: Result of Placing Item B and the Remaining Spaces.

Place B (30x30) in the top-left area (50x50). The remaining space is split into:

- Right Node: (20x30)
- Bottom node: (50x20)

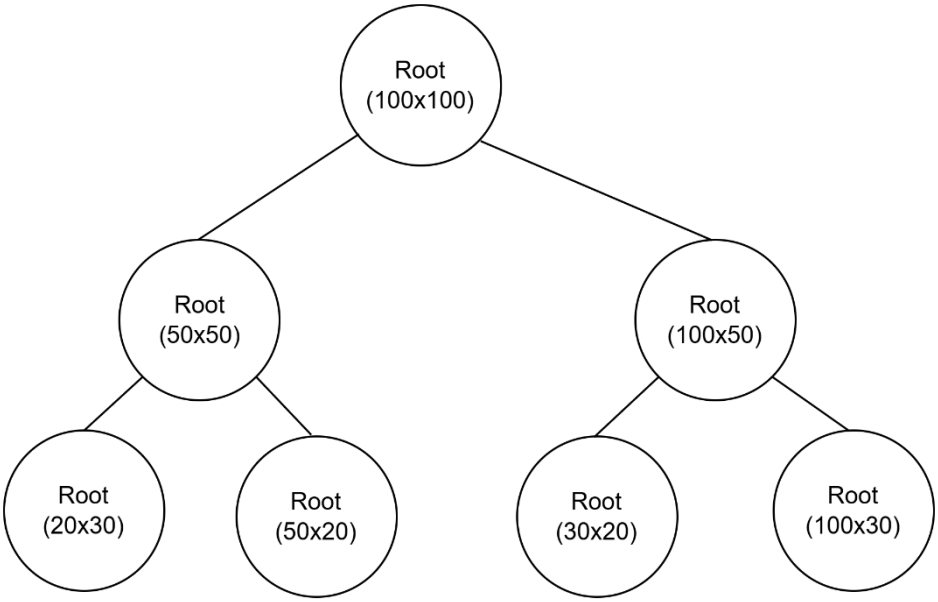



Figure 2. 35: Binary Tree.

5. Place Item D (10x10)

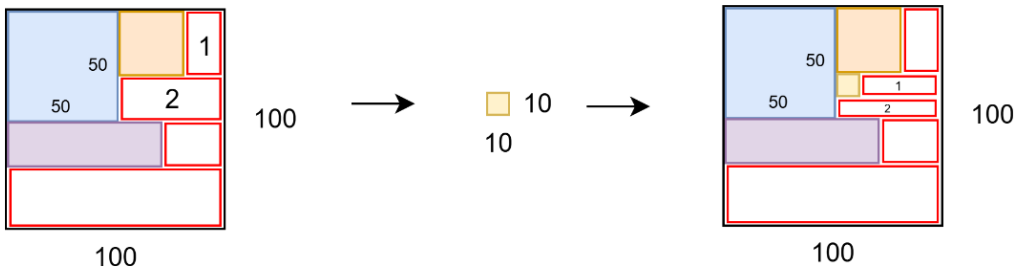



Figure 2. 36: Result of Placing Item D and the Remaining Spaces.

Item D (10x10) is placed inside the remaining (50x20) space. The remaining space is split into:

- Right Node: (40x10)
- Bottom Node: (50x10)

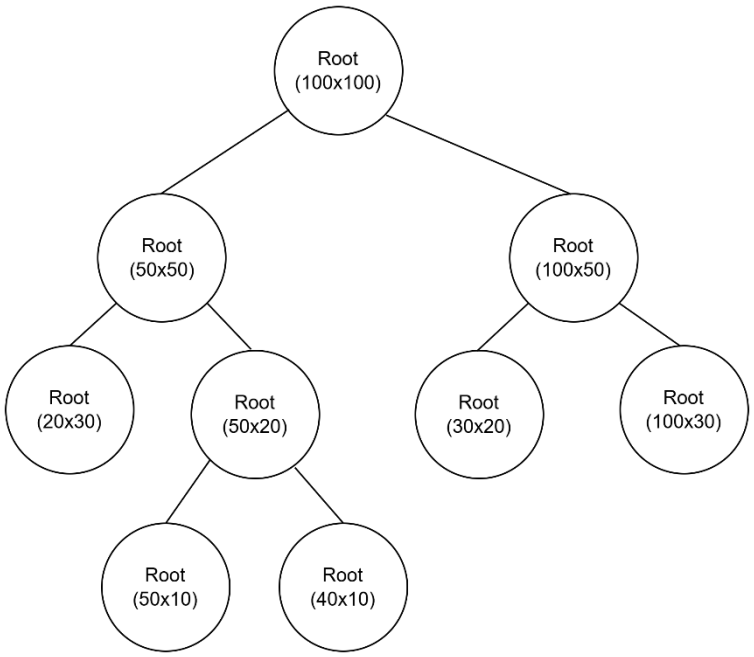



Figure 2. 37: Binary Tree.

## 2.7 Summary

After reviewing two algorithms, the Binary Tree Bin Packing Algorithm is chosen for implementation because it is easier to understand and implement, especially for scenarios involving only rectangular or square-shaped items. This algorithm uses a simple binary tree structure to divide space and place items efficiently, making it highly suitable for structured and grid-like packing tasks such as cargo load planning.

In contrast, the Biased Random Key Genetic Algorithm (BRKGA) is more complex, requiring the design of genetic encoding, fitness functions, and evolutionary operators. While BRKGA is powerful for solving a wide range of optimization problems and may achieve better results in some cases, it is computationally intensive and less straightforward to implement. Given the scope of this project and the need for simplicity and clarity, the Binary Tree Bin Packing Algorithm is the more practical and effective choice.

# CHAPTER 3

# METHODOLOGY AND WORK PLAN

## 3.1 Introduction

This chapter outlines the methods and tools used in the development of the proposed system. It covers the selected software development methodology, description of the tools used, and project planning for the entire project. A detailed Work Breakdown Structure (WBS) and Gantt Chart are also included to illustrate the project timeline and task management.

## 3.2 Software Development Methodology

The combination of throwaway prototyping and incremental development methodologies was selected for this system, as previously discussed in Section 1.5 (Proposed Solution). Since there is lack of fully defined user requirements, early developing an initial prototype helps to identify and refine the main features, while incremental development allows the system incrementally building feature by feature. This approach provide flexibility to changes in design and feature development.

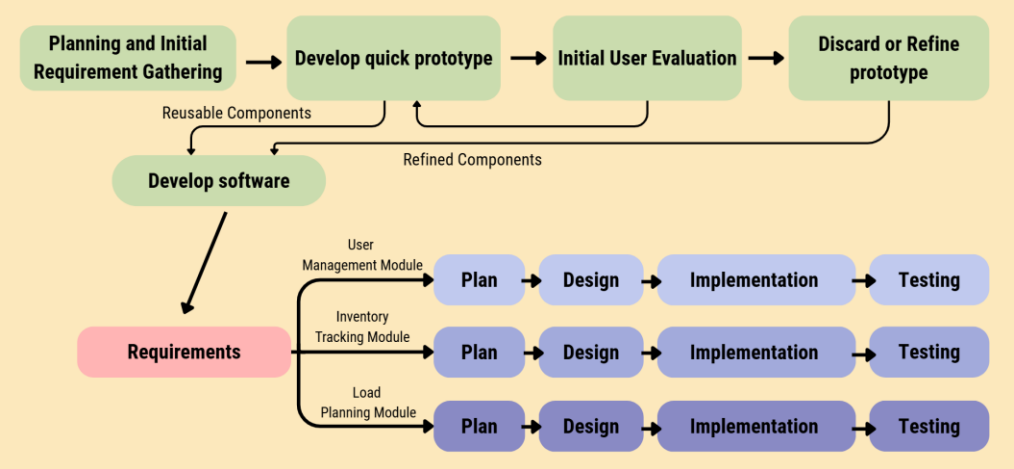



Figure 3. 1: Throwaway Prototyping with Incremental Model Development Methodologies.

### 3.2.1 Throwaway prototyping

Throwaway prototyping was used to address unclear and evolving requirements by rapidly creating early disposable versions of the application. This approach encourages experimentation and helps stakeholders, especially non-technical users to visualize and interact with the system during interviews. It reduces misunderstandings and supports more accurate feedback, ultimately helping to refine the software's final functionality before full-scale development begins. The overall process including planning, developing quick prototypes, gathering feedback, and refining features is outlined in the steps below.

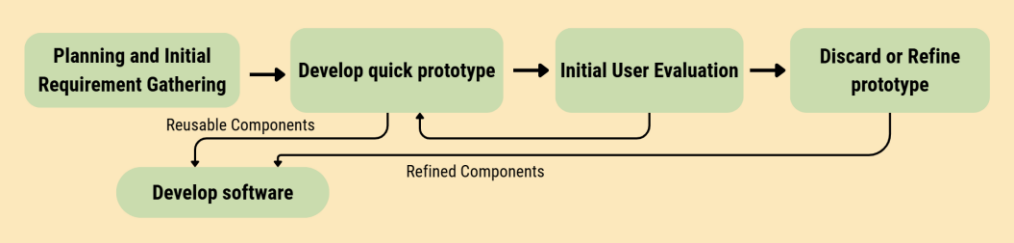



Figure 3. 2: Throwaway Prototyping Phases.

#### 3.2.1.1 Planning and Initial Requirement Gathering

The first step of throwaway prototyping is to gain a basic understanding of what the app should do, who it's for, what problems it solves and how to implement it. This phase begins by identifying problems within the current inventory tracking and load planning processes, particularly those relying on manual methods commonly used by small and medium-sized enterprises. Therefore, problem statements such as inaccurate record-keeping, lack of automation and time-consuming are identified. From this understanding, the project's goals and specific objectives are defined to guide solution development. Besides, a review of relevant literature and analysis of existing applications were conducted. Through hands-on experience with existing applications, the main functions were identified, and some additional features were proposed to make the application more acceptable to small and medium-sized enterprises. The project scope and limitations are also outlined, focusing on SME needs and constraints such as support for rectangle-shaped goods only.

Finally, application modules are identified, and functional and non-functional requirements are elicited.

#### 3.2.1.2 Develop a Quick Prototype

After defining the initial functionality of the application, the first version of the prototype was created using the Axure RP so that users without an IT background can gain a more intuitive understanding of the software's functionality and provide feedback. Based on the identified functionalities, prototype screens for 3 main modules were created, including user login and registration, inventory management via barcode scanning, and load planning. The designed screens are then linked together that allows users to navigate and interact like a real application. Once the complete functional prototype is created, an initial review is conducted through a self-evaluation to check the main usability of the main feature and identify the missing or incomplete components. Finally, the necessary improvements are being made to enhance the quality of prototypes. This refined prototype is then prepared for the next user feedback collection phase.

#### 3.2.1.3 Prototype Review and Get Feedback

After building the initial throwaway prototype, the next step is to collect feedback to evaluate the app's usability and overall concept of the application. This phase combined informal user testing and one-on-one interviews with experienced users of the relevant software. Target users, typically representatives from small and medium-sized enterprises, who were identified and selected for interviews to ensure that the information obtained was relevant to the project. Therefore, the interview was conducted with the Assistant Manager of the Logistics Department at a manufacturing company. The reason for the selection is that she has nearly 15 years of experience in logistics, which means she can provide expert insights into the load planning functionality. The interview was semi-structured, combining prepared questions with open-ended discussion, and covered the key features such as barcode scanning, inventory tracking process, and load planning interface.

In addition to the interview, potential end users have the opportunity to test prototypes that include the main features in order to understand

firsthand how users react and identify areas for improvement can be identified before moving into structured development. Informal testing was carried out with the interviewee to observe how she interacted with the prototype. She was asked to complete some basic tasks such as generating a load plan using container sizes and cargoes provided in the software. While experiencing the prototype, she made some suggestions about the user interface. For example, she pointed out that the color-coded cargo blocks appeared visually appealing within the mobile application. However, it becomes ineffective when exported or printed as a PDF, because companies often print documents in black and white rather than in colour to save costs. She recommended using text-based labels directly on the cargo blocks instead, to ensure that printed versions remain readable.

In conclusion, the feedback revealed that while the prototype's core concept was solid, several usability improvements were still needed to guide the transition to structured development.

#### 3.2.1.4 Discard or Refine the Prototype

After the feedback collection phase, the next step was to evaluate the throwaway prototype to determine whether it should be refined or discarded. Firstly, review the user comments in detail to find out where the prototype met expectations and where it fell short. Based on the user feedback obtained from the user interactions and testing, decisions are made. Some parts of the prototype were discarded, while others were enhanced and integrated into the next development phase. For example, the feature of using color-code to differentiate cargo groups was discarded, while using text-based labels placed on top of the cargoes feature will be refined. The use of color-coded cargo groups was found to be ineffective when printed in black-and-white. This is because most companies will print documents in black and white to save costs. No matter how beautiful it looks on the phone, it will be difficult for staff to distinguish between different types of goods. Additionally, screen layouts, navigation flows and functional requirements are updated accordingly. The use case diagrams and descriptions were updated to match the revised system functionality. Rather than discarding the prototype entirely, it was improved and adjusted based on the feedback gathered.

### 3.2.2 Incremental Process Model

After analysing the feedback, the project transitioned from the prototyping phase to a more structured development approach. The feedback gained from early user testing, interviews, and prototype refinement helped shape a clearer understanding of user expectations and requirements. Formal software development begins and each increment in the model follows the Software Development Life Cycle (SDLC), including Planning, Design, Implementation and Testing.

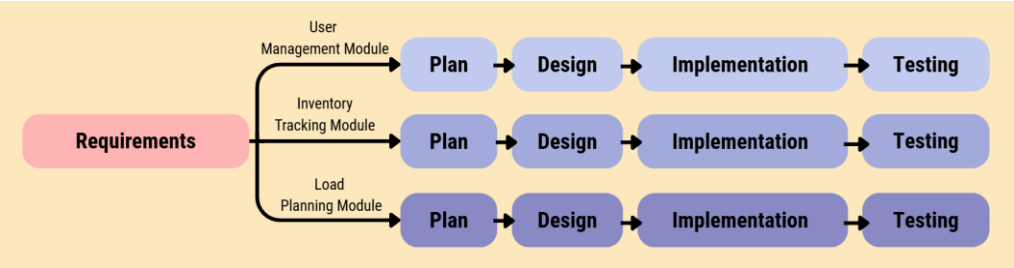



Figure 3. 3: Incremental Process Model Phases.

#### 3.2.2.1 First Increment

The first increment focused on implementing the User Management Module. The main objective of this phase was to develop a basic registration and login system to allow users to securely access the application. During the planning and analysis stage, the user requirements for authentication were defined, and Firebase Authentication was selected as the service to handle account creation and verification. In the design phase, user interfaces for the registration and login screens were created using React Native, ensuring a simple and user-friendly layout. The implementation was carried out in Visual Studio Code, with Firebase providing real-time authentication and data storage support. The key features developed included user account registration, secure login functionality, and session handling. Finally, functional testing was conducted by creating new accounts, attempting logins with valid and invalid credentials, and verifying that only registered users could successfully access the system's features.

#### 3.2.2.2 Second Increment

The second increment focused on developing the Inventory Tracking Module, which allows users to manage stock efficiently through barcode scanning and record inventory in and out activities with real-time updates. During the planning and analysis stage, the flow of inventory operations was defined, including how users would scan items, log stock movements, and monitor inventory levels. The design phase involved creating screens for barcode scanning, displaying item information, and managing inventory status. The interaction flow was also outlined to allow users to either check in or check out stock by scanning an item's barcode using the device camera. In the implementation phase, the react-native-camera library was used to enable barcode scanning, while the system checked database information against the scanned data records stored in Firebase. Real-time updates to stock quantities and activity logs were also implemented to maintain up-to-date inventory information. During the testing phase of the Inventory Tracking Module, several functional tests was carried out to ensure the system successfully capture and process barcode data, validate it against the database, and update inventory records correctly. The key focus was to verify that barcode scanning worked reliably and that stock movements were accurately reflected in the system.

#### 3.2.2.3 Third Increment

In the third increment, focused on the development of the Load Planning Module, which allows the users to optimize the arrangement of cargo within transportation containers or lorry. The goal was to generate an efficient load plan based on cargo dimensions, weight, and available container capacity to minimize unused space. During the planning and analysis stage, suitable algorithms for solving space allocation problems were reviewed, and the Binary Tree Bin Packing Algorithm was selected for implementation. This algorithm operates by recursively dividing available space into smaller sections using a binary tree structure. Each item is placed in the most optimal position, and the remaining space is split for future item placement.

During the design phase, the user interface screens were designed to guide users through the load planning process. This includes the screens for creating and editing load plans, adding custom cargo or container specifications, viewing load plan history, managing checklists, and generating printable PDF reports.

The implementation phase focus on integrating the load planning logic within the React Native application. The Binary Tree Bin Packing Algorithm was developed in JavaScript to process cargo details such as length, width, height, and quantity against the container's internal dimensions. Based on this input, the algorithm generated a placement layout that included position coordinates, cargo grouping, and overall space utilization. To enhance usability, a drag-and-drop feature was also implemented, allowing users to manually adjust cargo positions within the container layout if required. This ensured flexibility by letting users refine the algorithm-generated plan. Cargo blocks were displayed with colour codes and text labels for easy identification during the loading process, and the finalized plan could be exported as a printable PDF report containing a QR code linked to the load plan history.

The unit testing of the Binary Tree Bin Packing Algorithm involved running various sets of sample cargo data and container dimensions through the algorithm. The testing process ensured that the algorithm correctly placed cargo items in the available space without overlapping and maximized space utilization as expected.

## 3.3 Development Tools

To develop an Android mobile application for the inventory management system with the key functionalities such as barcode scanning for tracking inventory movement and load planning, several development tools are utilized throughout the implementation of the project.

### 3.3.1 Axure RP

Axure RP is the design tool used to create user interface prototypes before development begins. Due to its various interactive elements, including drag-and-drop widgets, dropdown menus, sliders, and dynamic panels, Axure was chosen for this project over alternative prototyping tools like Figma (Varun

Saharawat, 2024). Axure is able to create "if-then" scenarios since it also supports conditional logic. By simulating real-world user interactions, these characteristics facilitate the design and refinement of complicated ideas. By using Axure to create a functional prototype that includes the main key features, users are able to better understand the purpose of the application at an early stage and provide feedback to modify the original concept, rather than ultimately submitting a version that does not meet user needs and may have defects.

### 3.3.2 Visual Studio Code

Visual Studio Code (VS Code) was used as the primary code editor due to its lightweight nature, robust extension support, and integration with React Native. Visual Studio Code was chosen because it supports almost all major programming languages, and in this project, it was mainly used for JavaScript development, covering both frontend interfaces and backend logic. The Firebase SDK and JavaScript APIs were integrated into the project to handle database and authentication functionalities. VS Code was also used to write Firebase service logic, including adding, retrieving, and updating data from Firebase Firestore, as well as implementing the load planning algorithm directly in JavaScript. Its debugging tools and version control integration further streamlined the development process.

### 3.3.3 Android Studio

Android Studio is the official integrated development environment (IDE) for Android app development. Android Studio was used in this project as the emulator, which allowed for testing the app on Android devices without needing physical devices. This helps to test the app’s functionality directly on a simulated device. Android Studio also provided access to the Android Software Development Kit (SDK), which is essential for building and packaging the mobile application. Besides, the tool set in Android Studio enabled the development of Android-specific features such as permissions for accessing the camera during barcode scanning operations.

### 3.3.4 React Native

React Native is a popular open-source framework for building mobile applications using JavaScript and React. The app's interface will build using reusable React components. Each component such as buttons, lists and input fields could be easily reused across different parts of the app, maintaining consistency and reducing development time. Besides, the critical feature for inventory management, barcode scanning, was integrated using third-party libraries such as react-native-camera and react-native-barcode-scanner. React Native allowed seamless integration of native camera functionality with JavaScript, enabling the app to scan barcodes for tracking inventory items.

### 3.3.5 Firebase

The Firebase was chosen instead of MySQL is because it offers real-time database that better supports the needs of this project. Cloud Firestore, Firebase's flexible and scalable NoSQL cloud database, was used to store and synchronize application data in real time. All inventory data such as items, quantities, barcodes and load planning information were stored in Firestore, enabling seamless updates and synchronization between devices. Firestore's real-time synchronization ensured that any updates, such as adding, editing, or removing inventory items, were instantly reflected on all users' devices without requiring manual refresh. In addition, Firebase Authentication was utilized to manage secure user registration and login, while Firebase SDKs and APIs provided straightforward integration with the React Native application.

### 3.3.6 react-native-camera

The react-native-camera library is a crucial component in enabling barcode scanning functionality within the mobile application. It provided access to the device's camera and allowed the app to capture barcode data efficiently, forming the core inventory tracking feature. The library supports scanning multiple types of barcodes such as Code128, QR, and EAN. It also integrated a camera view directly in the app where users could scan the barcode within a predefined frame. Once a barcode was detected, the app retrieved the corresponding item information from Firebase and updated the inventory.

## 3.4 Work Breakdown Structure (WBS)

0.0 Inventory Tracking System with Barcode Scanning and Load Planning

1.0 Planning and Initial Requirement Gathering

- 1.1 Identify Problem
  - 1.1.1 Identify issues in current inventory and load planning process
    - 1.1.1.1 Analyze manual inventory methods
    - 1.1.1.2 Analyze current load planning techniques
  - 1.1.2 Develop problem statements
    - 1.1.2.1 Summarize inventory problems
    - 1.1.2.2 Summarize load planning problems
- 1.2 Identify project goals and objectives
  - 1.2.1 Define overall project goals
  - 1.2.2 Define specific objectives
- 1.3 Propose project solution
  - 1.3.1 Research similar solutions
  - 1.3.2 Compare similar solutions
    - 1.3.2.1 Analyse features offered
    - 1.3.2.2 Analyse pricing and suitability for SMEs
  - 1.3.3 Sketch flow of proposed solution
    - 1.3.3.1 Create flowchart of inventory tracking process
    - 1.3.3.2 Create flowchart of load planning process
- 1.4 Propose project approach
  - 1.4.1 Select software development methodology
    - 1.4.1.1 Choose Throwaway Prototyping for early design
    - 1.4.1.2 Choose Incremental Process Model for development phase
  - 1.4.2 Justify methodology choice
    - 1.4.2.1 Advantages of Throwaway Prototyping
    - 1.4.2.2 Advantages of Incremental Process
- 1.5 Define project scope and limitation
  - 1.5.1 Define application scope
    - 1.5.1.1 Focus on small and medium enterprises
    - 1.5.1.2 Focus on rectangular or square-shaped goods

1.5.2 Identify limitations

1.5.2.1 No support for irregularly shaped cargo

1.5.2.2 No complex weight and fragility handling

1.5.3 Identify required modules

1.5.3.1 Analyse User Management module

1.5.3.2 Analyse Inventory Tracking via Barcode Scanning module

1.5.3.3 Analyse Load Planning module

1.6 Gathering requirements

1.6.1 Review existing applications

1.6.1.1 Identify key features

1.6.1.2 Walkthrough of load planning processes

1.6.1.3 Identify pros and cons of each application

1.6.1.4 Compare and tabulate results

1.7 Literature Review

1.7.1 Study on inventory management process

1.7.2 Study on load planning algorithms

1.7.2.1 Analyse Biased Random Key Genetic Algorithm with Placement Procedure Heuristic

1.7.2.2 Analyse Binary Tree Bin Packing Algorithm

1.8 Requirements elicitation

1.8.1 Define functional requirements

1.8.2 Define non-functional requirements

1.8.3 Develop use case diagram

1.8.4 Write use case descriptions

1.8.5 Design interface flow diagram

1.9 Project scheduling

1.9.1 Prepare work break down structure (WBS)

1.9.1.1 Identify the main activities

1.9.1.2 Break down the activities into smaller tasks

1.9.2 Prepare Gantt Chart

1.9.2.1 Define task start and end dates

1.9.2.2 Identify task dependencies

1.9.2.3 Create Gantt Chart

2.0 Prototype Development

- 2.1 Define initial functionality
- 2.2 Select prototyping tool
- 2.2 Design initial prototype
  - 2.2.1 Sketch basic inventory tracking flow
  - 2.2.2 Sketch basic load planning flow
- 2.3 Build prototype screens
  - 2.3.1 Design login and registration process screens
  - 2.3.2 Design inventory tracking process screens
  - 2.3.3 Design load planning process screens
- 2.4 Link screens and define interactions
  - 2.4.1 Create navigation flows between screens
- 2.5 Review and finalize throwaway prototype
  - 2.5.1 Perform self-evaluation of usability
    - 2.5.1.1 Test core features
    - 2.5.1.2 Check user interface navigation and flow
    - 2.5.1.3 Identify missing functions
    - 2.5.1.4 Refine prototype
  - 2.5.2 Prepare prototype for user feedback

3.0 Prototype Review and Get Feedback

- 3.1 Perform Interviews
  - 3.1.1 Identify and select target users for interviews
  - 3.1.2 Prepare interview questions
  - 3.1.3 Conduct interviews with users
  - 3.1.4 Analyze feedback from users
- 3.2 Conduct Informal User Testing
  - 3.2.1 Prepare testing tasks
  - 3.2.2 Conduct testing sessions
    - 3.2.2.1 Observe user interaction
    - 3.2.2.2 Record user feedback
  - 3.2.3 Analyze test results
    - 3.2.3.1 Identify common issues and suggestions

4.0 Discard or Refine Prototype

- 4.1 Evaluate feedback

4.1.1 Review user comments and observations

4.1.2 Summarize key usability problems

4.2 Decide action for prototype

4.2.1 Identify features to discard

4.2.2 Identify features to refine

4.3 Update prototype design

4.3.1 Modify screens and navigation based on feedback

4.3.2 Modify function requirements

4.3.3 Modify use case diagram

4.3.4 Modify use case description

5.0 Incremental Development

5.1 First Increment

5.1.1 Planning and Analysis

5.1.1.1 Define user authentication requirements

5.1.1.2 Select authentication method

5.1.2 Design

5.1.2.1 Design login account and register new user screens

5.1.2.2 Design basic navigation flow after login

5.1.3 Implementation

5.1.3.1 Build registration UI

5.1.3.2 Build login UI

5.1.3.3 Implement user account creation with Firebase

5.1.3.4 Implement login functionality with Firebase Authentication

5.1.3.5 Integrate session handling for logged-in users

5.1.4 Testing

5.1.4.1 Test account registration with valid/invalid inputs

5.1.4.2 Test login with correct and incorrect credentials

5.1.4.3 Test access to system features after successful login

5.1.4.4 Test logout functionality

5.2 Second Increment

5.2.1 Planning and Analysis

5.2.1.1 Analyze requirements for inventory listing, filtering, and searching

5.2.1.2 Define inventory item data structure

5.2.1.3 Plan database structure for inventory management

5.2.1.4 Define barcode scanning requirements

5.2.1.5 Define stock update process via scanning

5.2.1.6 Plan alert conditions for low stock and unmatched barcodes

5.2.1.7 Define requirements for inventory summary report generation

5.2.2 Design

5.2.2.1 Design screens

5.2.2.1.1 Inventory list screen

5.2.2.1.2 Add and Edit inventory item screens

5.2.2.1.3 Barcode scanner interface

5.2.2.1.4 Alerts and Notifications indicators

5.2.2.1.5 Inventory Summary Report screen.

5.2.3 Implementation

5.2.3.1 Implement inventory item listing

5.2.3.2 Implement filtering and searching features

5.2.3.3 Implement add new item feature

5.2.3.4 Implement update item details feature

5.2.3.5 Integrate barcode scanning feature using device camera

5.2.3.6 Implement stock quantity updates via scanning

5.2.3.7 Implement delete item feature

5.2.3.8 Implement low stock alert indicator

5.2.3.9 Implement unmatched barcode alert

5.2.3.10 Implement inventory summary report generation

5.2.4 Testing

5.2.4.1 Test inventory list viewing, filtering, and searching

5.2.4.2 Test barcode scanning functionality

5.2.4.3 Test adding, updating, and deleting items

5.2.4.4 Test stock quantity updates via scanning

5.2.4.5 Test low stock alert triggering

5.2.4.6 Test unmatched barcode handling

5.3 Third Increment

5.3.1 Planning and Analysis

5.3.1.1 Define cargo and container attributes

5.3.1.2 Plan database structure for storing load plans

5.3.1.3 Define checklist workflow and data structure

5.3.1.4 Define load plan update mechanism

5.3.1.5 Plan the generation of printable PDF reports

5.3.1.6 Define QR code generation and retrieval process

5.3.2 Design screen

5.3.2.1 Load plan creation screen

5.3.2.2 Add custom cargo and container screen

5.3.2.3 Load plan editing screen with drag-and-drop interface

5.3.2.4 PDF layout for printable load plan report

5.3.2.5 Load plan history screen

5.3.2.6 Cargo arrangement checklist screen

5.3.3 Implementation

5.3.3.1 Implement load plan generation

5.3.3.2 Implement add custom containers and cargo size feature

5.3.3.3 Implement drag-and-drop adjustment feature

5.3.3.4 Implement PDF report generation with QR code embedded

5.3.3.5 Implement QR code scanning

5.3.3.6 Implement checklist functionality

5.3.4 Testing

5.3.4.1 Test load plan creation

5.3.4.2 Test drag-and-drop adjustments of cargo

5.3.4.3 Test scanning QR codes to view correct load plan history

5.3.4.4 Test checklist functionality for checking off cargo items

## 3.5 Gantt Chart

### 3.5.1 Overview of Project Timeline

Figure 3. 4: Gantt Chart for Overall Project.

### 3.5.2 Planning & Initial Requirement Gathering

| ID | Task Name | Duration | Start | Finish |
|---|---|---|---|---|
| 2 | **1.0 Planning and Initial Requirement Gathering** | **45 days** | **Mon 25/2/10** | **Fri 25/3/28** |
| 3 | **1.1 Identify Problem** | **6 days** | **Mon 25/2/10** | **Sat 25/2/15** |
| 4 | **1.1.1 Identify issues in current inventory and load planning process** | **3 days** | **Mon 25/2/10** | **Wed 25/2/12** |
| 5 | 1.1.1.1 Analyze manual inventory methods | 2 days | Mon 25/2/10 | Tue 25/2/11 |
| 6 | 1.1.1.2 Analyze current load planning techniques | 2 days | Tue 25/2/11 | Wed 25/2/12 |
| 7 | **1.1.2 Develop problem statements** | **4 days** | **Wed 25/2/12** | **Sat 25/2/15** |
| 8 | 1.1.2.1 Summarize inventory problems | 2 days | Wed 25/2/12 | Thu 25/2/13 |
| 9 | 1.1.2.2 Summarize load planning problems | 3 days | Thu 25/2/13 | Sat 25/2/15 |
| 10 | **1.2 Identify project goals and objectives** | **3 days** | **Sat 25/2/15** | **Mon 25/2/17** |
| 11 | 1.2.1 Define overall project goals | 2 days | Sat 25/2/15 | Sun 25/2/16 |
| 12 | 1.2.2 Define specific objectives | 2 days | Sun 25/2/16 | Mon 25/2/17 |
| 13 | **1.3 Propose project solution** | **6 days** | **Tue 25/2/18** | **Sun 25/2/23** |
| 14 | 1.3.1 Research similar solutions | 3 days | Tue 25/2/18 | Thu 25/2/20 |
| 15 | **1.3.2 Compare similar solutions** | **2 days** | **Thu 25/2/20** | **Fri 25/2/21** |
| 16 | 1.3.2.1 Analyze features offered | 1 day | Thu 25/2/20 | Thu 25/2/20 |
| 17 | 1.3.2.2 Analyze pricing and suitability for SMEs | 2 days | Thu 25/2/20 | Fri 25/2/21 |
| 18 | **1.3.3 Sketch flow of proposed solution** | **3 days** | **Fri 25/2/21** | **Sun 25/2/23** |
| 19 | 1.3.3.1 Create flowchart of inventory tracking proces | 3 days | Fri 25/2/21 | Sun 25/2/23 |
| 20 | 1.3.3.2 Create flowchart of load planning process | 3 days | Fri 25/2/21 | Sun 25/2/23 |
| 21 | **1.4 Propose project approach** | **6 days** | **Sat 25/2/22** | **Thu 25/2/27** |
| 22 | **1.4.1 Select software development methodology** | **4 days** | **Sat 25/2/22** | **Tue 25/2/25** |
| 23 | 1.4.1.1 Choose Throwaway Prototyping for early design | 4 days | Sat 25/2/22 | Tue 25/2/25 |
| 24 | 1.4.1.2 Choose Incremental Process Model for development phase | 4 days | Sat 25/2/22 | Tue 25/2/25 |
| 25 | **1.4.2 Justify methodology choice** | **4 days** | **Mon 25/2/24** | **Thu 25/2/27** |
| 26 | 1.4.2.1 Advantages of Throwaway Prototyping | 4 days | Mon 25/2/24 | Thu 25/2/27 |
| 27 | 1.4.2.2 Advantages of Incremental Process | 4 days | Mon 25/2/24 | Thu 25/2/27 |
| 28 | **1.5 Define project scope and limitation** | **4 days** | **Fri 25/2/28** | **Mon 25/3/3** |
| 29 | **1.5.1 Define application scope** | **1 day** | **Fri 25/2/28** | **Fri 25/2/28** |
| 30 | 1.5.1.1 Focus on small and medium enterprises | 1 day | Fri 25/2/28 | Fri 25/2/28 |
| 31 | 1.5.1.2 Focus on rectangular or square-shaped goo | 1 day | Fri 25/2/28 | Fri 25/2/28 |
| 32 | **1.5.2 Identify limitations** | **1 day** | **Sat 25/3/1** | **Sat 25/3/1** |
| 33 | 1.5.2.1 No support for irregularly shaped cargo | 1 day | Sat 25/3/1 | Sat 25/3/1 |
| 34 | 1.5.2.2 No complex weight and fragility handling | 1 day | Sat 25/3/1 | Sat 25/3/1 |
| 35 | **1.5.3 Identify required modules** | **2 days** | **Sun 25/3/2** | **Mon 25/3/3** |
| 36 | 1.5.3.1 Analyze User Management module | 1 day | Sun 25/3/2 | Sun 25/3/2 |
| 37 | 1.5.3.2 Analyze Inventory Tracking via Barcode Scanning module | 2 days | Sun 25/3/2 | Mon 25/3/3 |
| 38 | 1.5.3.3 Analyze Load Planning module | 2 days | Sun 25/3/2 | Mon 25/3/3 |
| 39 | **1.6 Gathering requirements** | **7 days** | **Tue 25/3/4** | **Tue 25/3/11** |
| 40 | **1.6.1 Review existing applications** | **7 days** | **Tue 25/3/4** | **Tue 25/3/11** |
| 41 | 1.6.1.1 Identify key features | 3 days | Tue 25/3/4 | Thu 25/3/6 |
| 42 | 1.6.1.2 Walkthrough of load planning processes | 2 days | Thu 25/3/6 | Fri 25/3/7 |
| 43 | 1.6.1.3 Identify pros and cons of each application | 2 days | Fri 25/3/7 | Sun 25/3/9 |
| 44 | 1.6.1.4 Compare and tabulate results | 2 days | Mon 25/3/10 | Tue 25/3/11 |
| 45 | **1.7 Literature Review** | **9 days** | **Mon 25/3/3** | **Wed 25/3/12** |
| 46 | 1.7.1 Study on inventory management process | 3 days | Mon 25/3/3 | Wed 25/3/5 |
| 47 | **1.7.2 Study on load planning algorithms** | **6 days** | **Thu 25/3/6** | **Wed 25/3/12** |
| 48 | 1.7.2.1 Analyze Biased Random Key Genetic Algorith | 5 days | Thu 25/3/6 | Tue 25/3/11 |
| 49 | 1.7.2.2 Analyze Binary Tree Bin Packing Algorithm | 3 days | Mon 25/3/10 | Wed 25/3/12 |
| 50 | **1.8 Requirements elicitation** | **10 days** | **Thu 25/3/13** | **Sun 25/3/23** |
| 51 | 1.8.1 Define functional requirements | 3 days | Thu 25/3/13 | Sat 25/3/15 |
| 52 | 1.8.2 Define non-functional requirements | 3 days | Thu 25/3/13 | Sat 25/3/15 |
| 53 | 1.8.3 Develop use case diagram | 2 days | Sat 25/3/15 | Sun 25/3/16 |
| 54 | 1.8.4 Write use case descriptions | 4 days | Sun 25/3/16 | Wed 25/3/19 |
| 55 | 1.8.5 Design interface flow diagram | 6 days | Mon 25/3/17 | Sun 25/3/23 |
| 56 | **1.9 Project scheduling** | **5 days** | **Mon 25/3/24** | **Fri 25/3/28** |
| 57 | **1.9.1 Prepare work break down structure (WBS)** | **3 days** | **Mon 25/3/24** | **Wed 25/3/26** |
| 58 | 1.9.1.1 Identify the main activities | 1 day | Mon 25/3/24 | Mon 25/3/24 |
| 59 | 1.9.1.2 Break down the activities into smaller tasks | 2 days | Tue 25/3/25 | Wed 25/3/26 |
| 60 | **1.9.2 Prepare Gantt Chart** | **2 days** | **Thu 25/3/27** | **Fri 25/3/28** |
| 61 | 1.9.2.1 Define task start and end dates | 1 day | Thu 25/3/27 | Thu 25/3/27 |
| 62 | 1.9.2.2 Identify task dependencies | 1 day | Fri 25/3/28 | Fri 25/3/28 |
| 63 | 1.9.2.3 Create Gantt Chart | 1 day | Fri 25/3/28 | Fri 25/3/28 |

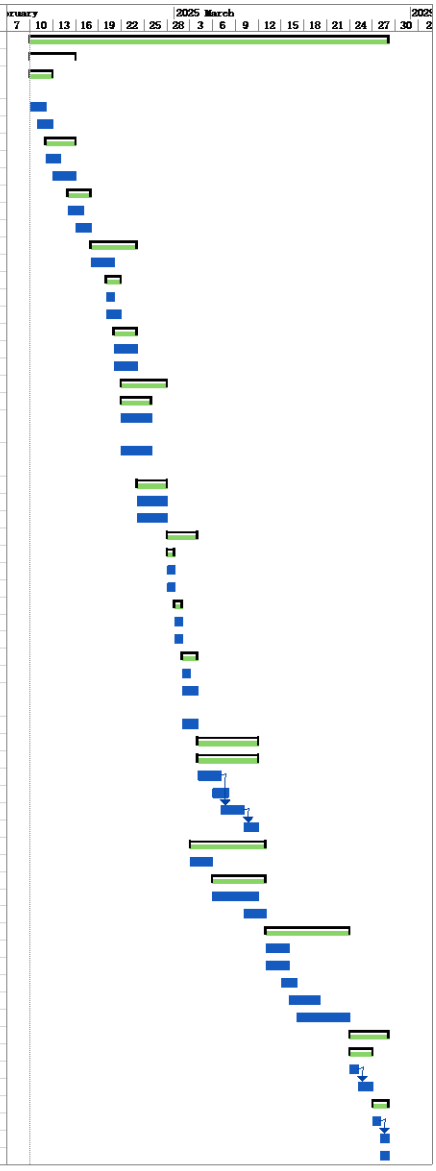



Figure 3. 5: Gantt Chart for Planning & Initial Requirement Gathering.

### 3.5.3 Prototype Development

| ID | Task Name | Duration | Start | Finish |
|---|---|---|---|---|
| 64 | **2.0 Prototype Development** | **13 days** | **Sat 25/3/29** | **Fri 25/4/11** |
| 65 | 2.1 Define initial functionality | 2 days | Sat 25/3/29 | Mon 25/3/31 |
| 66 | 2.2 Select prototyping tool | 1 day | Mon 25/3/31 | Mon 25/3/31 |
| 67 | **2.3 Design initial prototype** | **2 days** | **Tue 25/4/1** | **Wed 25/4/2** |
| 68 | 2.3.1 Sketch basic inventory tracking flow | 1 day | Tue 25/4/1 | Tue 25/4/1 |
| 69 | 2.3.2 Sketch basic load planning flow | 2 days | Tue 25/4/1 | Wed 25/4/2 |
| 70 | **2.4 Build prototype screens** | **5 days** | **Wed 25/4/2** | **Sun 25/4/6** |
| 71 | 2.4.1 Design login and registration process screens | 2 days | Wed 25/4/2 | Thu 25/4/3 |
| 72 | 2.4.2 Design inventory tracking process screens | 2 days | Thu 25/4/3 | Fri 25/4/4 |
| 73 | 2.4.3 Design load planning process screens | 3 days | Fri 25/4/4 | Sun 25/4/6 |
| 74 | **2.5 Link screens and define interactions** | **2 days** | **Sun 25/4/6** | **Mon 25/4/7** |
| 75 | 2.5.1 Create navigation flows between screens | 2 days | Sun 25/4/6 | Mon 25/4/7 |
| 76 | **2.6 Review and finalize throwaway prototype** | **3 days** | **Tue 25/4/8** | **Thu 25/4/10** |
| 77 | **2.6.1 Perform self-evaluation of usability** | **3 days** | **Tue 25/4/8** | **Thu 25/4/10** |
| 78 | 2.6.1.1 Test core features | 2 days | Tue 25/4/8 | Wed 25/4/9 |
| 79 | 2.6.1.2 Check user interface navigation and flow | 2 days | Wed 25/4/9 | Thu 25/4/10 |
| 80 | 2.6.1.3 Identify missing functions | 3 days | Tue 25/4/8 | Thu 25/4/10 |
| 81 | 2.6.1.4 Refine prototype | 3 days | Tue 25/4/8 | Thu 25/4/10 |
| 82 | 2.6.2 Prepare prototype for user feedback | 1 day | Thu 25/4/10 | Thu 25/4/10 |

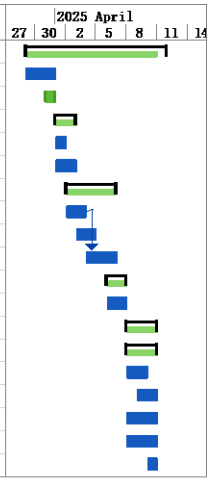



Figure 3. 6: Gantt Chart for Prototype Development.

### 3.5.4 Prototype Review and Get Feedback

| ID | Task Name | Duration | Start | Finish |
|---|---|---|---|---|
| 83 | **3.0 Prototype Review and Get Feedback** | **7 days** | **Sat 25/4/12** | **Sun 25/4/20** |
| 84 | **3.1 Perform Interviews** | **3 days** | **Sat 25/4/12** | **Tue 25/4/15** |
| 85 | 3.1.1 Identify and select target users for interviews | 1 day | Sat 25/4/12 | Sat 25/4/12 |
| 86 | 3.1.2 Prepare interview questions | 2 days | Sat 25/4/12 | Mon 25/4/14 |
| 87 | 3.1.3 Conduct interviews with users | 1 day | Mon 25/4/14 | Mon 25/4/14 |
| 88 | 3.1.4 Analyze feedback from users | 3 days | Tue 25/4/15 | Thu 25/4/17 |
| 89 | **3.2 Conduct Informal User Testing** | **5 days** | **Mon 25/4/14** | **Fri 25/4/18** |
| 90 | 3.2.1 Prepare testing tasks | 1 day | Mon 25/4/14 | Mon 25/4/14 |
| 91 | **3.2.2 Conduct testing sessions** | **1 day** | **Mon 25/4/14** | **Mon 25/4/14** |
| 92 | 3.2.2.1 Observe user interaction | 1 day | Mon 25/4/14 | Mon 25/4/14 |
| 93 | 3.2.2.2 Record user feedback | 1 day | Mon 25/4/14 | Mon 25/4/14 |
| 94 | **3.2.3 Analyze test results** | **4 days** | **Tue 25/4/15** | **Fri 25/4/18** |
| 95 | 3.2.3.1 Identify common issues and suggestions | 4 days | Tue 25/4/15 | Fri 25/4/18 |

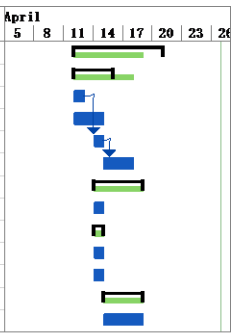



Figure 3. 7: Gantt Chart for Prototype Review and Get Feedback.

### 3.5.5 Discard or Refine Prototype

| ID | Task Name | Duration | Start | Finish |
|---|---|---|---|---|
| 96 | **4.0 Discard or Refine Prototype** | **31 days** | **Mon 25/4/21** | **Thu 25/5/29** |
| 97 | **4.1 Evaluate feedback** | **4 days** | **Mon 25/4/21** | **Thu 25/4/24** |
| 98 | 4.1.1 Review user comments and observations | 2 days | Mon 25/4/21 | Tue 25/4/22 |
| 99 | 4.1.2 Summarize key usability problems | 3 days | Tue 25/4/22 | Thu 25/4/24 |
| 100 | **4.2 Decide action for prototype** | **3 days** | **Mon 25/4/21** | **Wed 25/4/23** |
| 101 | 4.2.1 Identify features to discard | 2 days | Mon 25/4/21 | Tue 25/4/22 |
| 102 | 4.2.2 Identify features to refine | 3 days | Mon 25/4/21 | Wed 25/4/23 |
| 103 | **4.3 Update prototype design** | **28 days** | **Thu 25/4/24** | **Thu 25/5/29** |
| 104 | 4.3.1 Modify screens and navigation | 14 days | Thu 25/4/24 | Mon 25/5/12 |
| 105 | 4.3.2 Modify function requirements | 4 days | Tue 25/5/13 | Fri 25/5/16 |
| 106 | 4.3.3 Modify use case diagram | 4 days | Mon 25/5/19 | Thu 25/5/22 |
| 107 | 4.3.4 Modify use case description | 6 days | Fri 25/5/23 | Thu 25/5/29 |

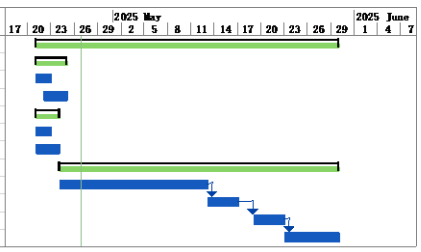
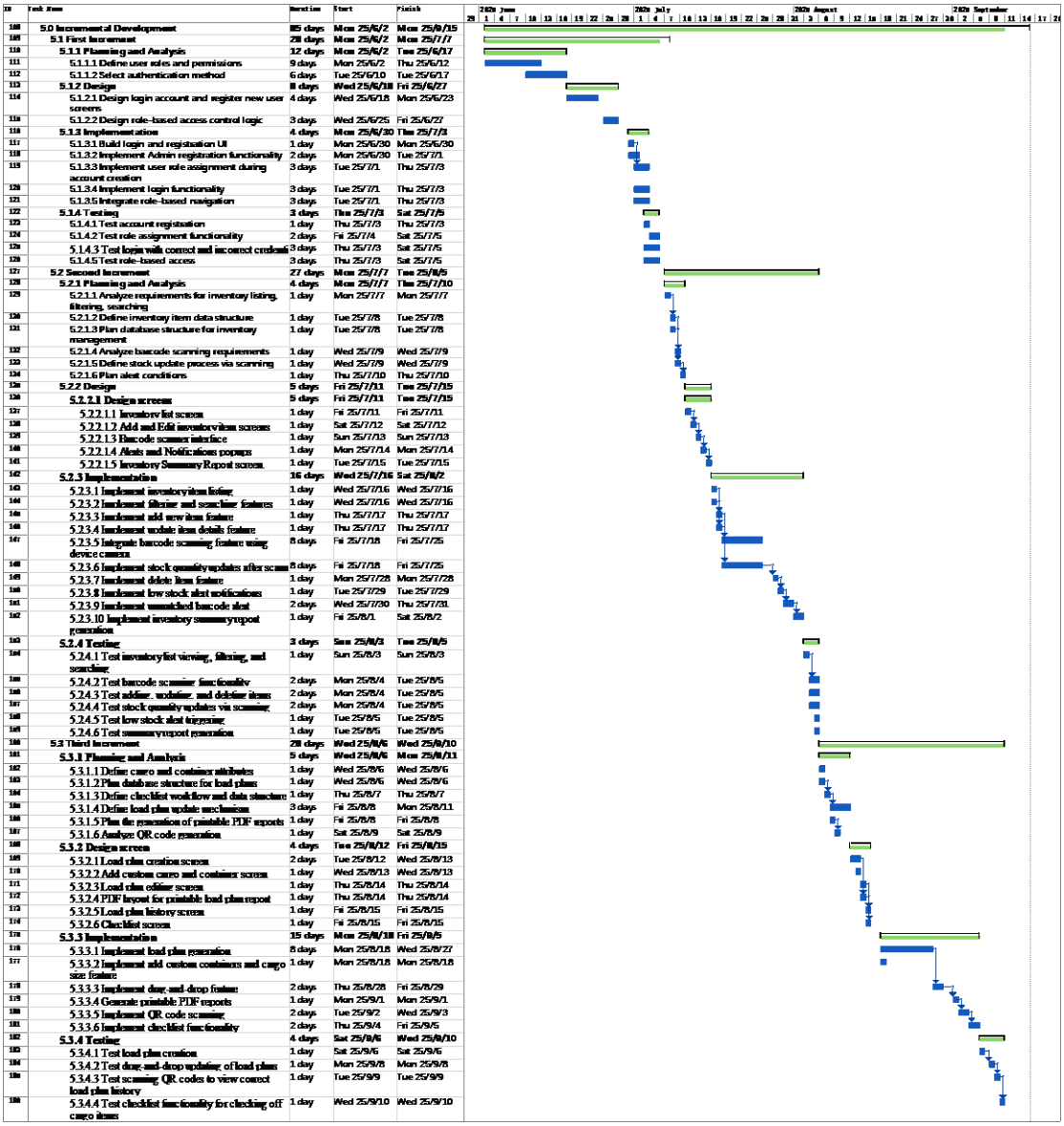



Figure 3. 8: Gantt Chart for Discard or Refine Prototype.

### 3.5.6 Incremental Development



| ID | Task Name | Duration | Start | Finish |
|---|---|---|---|---|
| 108 | 5.0 Incremental Development | 85 days | Mon 25/6/2 | Mon 25/9/15 |
| 109 | 5.1 First Increment | 28 days | Mon 25/6/2 | Mon 25/7/7 |
| 110 | 5.1.1 Planning and Analysis | 12 days | Mon 25/6/2 | Tue 25/6/17 |
| 111 | 5.1.1.1 Define user roles and permissions | 9 days | Mon 25/6/2 | Thu 25/6/12 |
| 112 | 5.1.1.2 Select authentication method | 6 days | Tue 25/6/10 | Tue 25/6/17 |
| 113 | 5.1.2 Design | 8 days | Wed 25/6/18 | Fri 25/6/27 |
| 114 | 5.1.2.1 Design login account and register new user screens | 4 days | Wed 25/6/18 | Mon 25/6/23 |
| 116 | 5.1.2.2 Design role-based access control logic | 3 days | Wed 25/6/25 | Fri 25/6/27 |
| 116 | 5.1.3 Implementation | 4 days | Mon 25/6/30 | Thu 25/7/3 |
| 117 | 5.1.3.1 Build login and registration UI | 1 day | Mon 25/6/30 | Mon 25/6/30 |
| 118 | 5.1.3.2 Implement Admin registration functionality | 2 days | Mon 25/6/30 | Tue 25/7/1 |
| 119 | 5.1.3.3 Implement user role assignment during account creation | 3 days | Tue 25/7/1 | Thu 25/7/3 |
| 120 | 5.1.3.4 Implement login functionality | 3 days | Tue 25/7/1 | Thu 25/7/3 |
| 121 | 5.1.3.5 Integrate role-based navigation | 3 days | Tue 25/7/1 | Thu 25/7/3 |
| 122 | 5.1.4 Testing | 3 days | Thu 25/7/3 | Sat 25/7/5 |
| 123 | 5.1.4.1 Test account registration | 1 day | Thu 25/7/3 | Thu 25/7/3 |
| 124 | 5.1.4.2 Test role assignment functionality | 2 days | Fri 25/7/4 | Sat 25/7/5 |
| 125 | 5.1.4.3 Test login with correct and incorrect credentials | 3 days | Thu 25/7/3 | Sat 25/7/5 |
| 126 | 5.1.4.5 Test role-based access | 3 days | Thu 25/7/3 | Sat 25/7/5 |
| 127 | 5.2 Second Increment | 27 days | Mon 25/7/7 | Tue 25/8/5 |
| 128 | 5.2.1 Planning and Analysis | 4 days | Mon 25/7/7 | Thu 25/7/10 |
| 129 | 5.2.1.1 Analyze requirements for inventory listing, filtering, searching | 1 day | Mon 25/7/7 | Mon 25/7/7 |
| 130 | 5.2.1.2 Define inventory item data structure | 1 day | Tue 25/7/8 | Tue 25/7/8 |
| 131 | 5.2.1.3 Plan database structure for inventory management | 1 day | Tue 25/7/8 | Tue 25/7/8 |
| 132 | 5.2.1.4 Analyze barcode scanning requirements | 1 day | Wed 25/7/9 | Wed 25/7/9 |
| 133 | 5.2.1.5 Define stock update process via scanning | 1 day | Wed 25/7/9 | Wed 25/7/9 |
| 134 | 5.2.1.6 Plan alert conditions | 1 day | Thu 25/7/10 | Thu 25/7/10 |
| 135 | 5.2.2 Design | 5 days | Fri 25/7/11 | Tue 25/7/15 |
| 136 | 5.2.2.1 Design screens | 5 days | Fri 25/7/11 | Tue 25/7/15 |
| 137 | 5.2.2.1.1 Inventory list screen | 1 day | Fri 25/7/11 | Fri 25/7/11 |
| 138 | 5.2.2.1.2 Add and Edit inventory item screens | 1 day | Sat 25/7/12 | Sat 25/7/12 |
| 139 | 5.2.2.1.3 Barcode scanner interface | 1 day | Sun 25/7/13 | Sun 25/7/13 |
| 140 | 5.2.2.1.4 Alerts and Notifications popups | 1 day | Mon 25/7/14 | Mon 25/7/14 |
| 141 | 5.2.2.1.5 Inventory Summary Report screen | 1 day | Tue 25/7/15 | Tue 25/7/15 |
| 142 | 5.2.3 Implementation | 16 days | Wed 25/7/16 | Sat 25/8/2 |
| 143 | 5.2.3.1 Implement inventory item listing | 1 day | Wed 25/7/16 | Wed 25/7/16 |
| 144 | 5.2.3.2 Implement filtering and searching features | 1 day | Wed 25/7/16 | Wed 25/7/16 |
| 145 | 5.2.3.3 Implement add new item feature | 1 day | Thu 25/7/17 | Thu 25/7/17 |
| 146 | 5.2.3.4 Implement update item details feature | 1 day | Thu 25/7/17 | Thu 25/7/17 |
| 147 | 5.2.3.5 Integrate barcode scanning feature using device camera | 8 days | Fri 25/7/18 | Fri 25/7/25 |
| 148 | 5.2.3.6 Implement stock quantity updates after scans | 8 days | Fri 25/7/18 | Fri 25/7/25 |
| 149 | 5.2.3.7 Implement delete item feature | 1 day | Mon 25/7/28 | Mon 25/7/28 |
| 150 | 5.2.3.8 Implement low stock alert notifications | 1 day | Tue 25/7/29 | Tue 25/7/29 |
| 151 | 5.2.3.9 Implement unmatched barcode alert | 2 days | Wed 25/7/30 | Thu 25/7/31 |
| 152 | 5.2.3.10 Implement inventory summary report generation | 1 day | Fri 25/8/1 | Sat 25/8/2 |
| 153 | 5.2.4 Testing | 3 days | Sun 25/8/3 | Tue 25/8/5 |
| 154 | 5.2.4.1 Test inventory list viewing, filtering, and searching | 1 day | Sun 25/8/3 | Sun 25/8/3 |
| 155 | 5.2.4.2 Test barcode scanner functionality | 2 days | Mon 25/8/4 | Tue 25/8/5 |
| 156 | 5.2.4.3 Test adding, updating, and deleting items | 2 days | Mon 25/8/4 | Tue 25/8/5 |
| 157 | 5.2.4.4 Test stock quantity updates via scanning | 2 days | Mon 25/8/4 | Tue 25/8/5 |
| 158 | 5.2.4.5 Test low stock alert triggering | 1 day | Tue 25/8/5 | Tue 25/8/5 |
| 159 | 5.2.4.6 Test summary report generation | 1 day | Tue 25/8/5 | Tue 25/8/5 |
| 160 | 5.3 Third Increment | 28 days | Wed 25/8/6 | Wed 25/9/10 |
| 161 | 5.3.1 Planning and Analysis | 5 days | Wed 25/8/6 | Mon 25/8/11 |
| 162 | 5.3.1.1 Define cargo and container attributes | 1 day | Wed 25/8/6 | Wed 25/8/6 |
| 163 | 5.3.1.2 Plan database structure for load plans | 1 day | Wed 25/8/6 | Wed 25/8/6 |
| 164 | 5.3.1.3 Define checklist workflow and data structure | 1 day | Thu 25/8/7 | Thu 25/8/7 |
| 165 | 5.3.1.4 Define load plan update mechanism | 3 days | Fri 25/8/8 | Mon 25/8/11 |
| 166 | 5.3.1.5 Plan the generation of printable PDF reports | 1 day | Fri 25/8/8 | Fri 25/8/8 |
| 167 | 5.3.1.6 Analyze QR code generation | 1 day | Sat 25/8/9 | Sat 25/8/9 |
| 168 | 5.3.2 Design screen | 4 days | Tue 25/8/12 | Fri 25/8/15 |
| 169 | 5.3.2.1 Load plan creation screen | 2 days | Tue 25/8/12 | Wed 25/8/13 |
| 170 | 5.3.2.2 Add custom cargo and container screen | 1 day | Wed 25/8/13 | Wed 25/8/13 |
| 171 | 5.3.2.3 Load plan edition screen | 1 day | Thu 25/8/14 | Thu 25/8/14 |
| 172 | 5.3.2.4 PDF layout for printable load plan report | 1 day | Thu 25/8/14 | Thu 25/8/14 |
| 173 | 5.3.2.5 Load plan history screen | 1 day | Fri 25/8/15 | Fri 25/8/15 |
| 174 | 5.3.2.6 Checklist screen | 1 day | Fri 25/8/15 | Fri 25/8/15 |
| 175 | 5.3.3 Implementation | 15 days | Mon 25/8/18 | Fri 25/9/5 |
| 176 | 5.3.3.1 Implement load plan generation | 8 days | Mon 25/8/18 | Wed 25/8/27 |
| 177 | 5.3.3.2 Implement add custom containers and cargo size feature | 1 day | Mon 25/8/18 | Mon 25/8/18 |
| 178 | 5.3.3.3 Implement drag-and-drop feature | 2 days | Thu 25/8/28 | Fri 25/8/29 |
| 179 | 5.3.3.4 Generate printable PDF reports | 1 day | Mon 25/9/1 | Mon 25/9/1 |
| 180 | 5.3.3.5 Implement QR code scanning | 2 days | Tue 25/9/2 | Wed 25/9/3 |
| 181 | 5.3.3.6 Implement checklist functionality | 2 days | Thu 25/9/4 | Fri 25/9/5 |
| 182 | 5.3.4 Testing | 4 days | Sat 25/9/6 | Wed 25/9/10 |
| 183 | 5.3.4.1 Test load plan creation | 1 day | Sat 25/9/6 | Sat 25/9/6 |
| 184 | 5.3.4.2 Test drag-and-drop updating of load plans | 1 day | Mon 25/9/8 | Mon 25/9/8 |
| 185 | 5.3.4.3 Test scanning QR codes to view correct load plan history | 1 day | Tue 25/9/9 | Tue 25/9/9 |
| 186 | 5.3.4.4 Test checklist functionality for checking off cargo items | 1 day | Wed 25/9/10 | Wed 25/9/10 |

Figure 3. 9: Gantt Chart for Overview of Incremental Development.

### 3.5.6.1 First Increment

| | Task Name | Duration | Start | Finish |
|---|---|---|---|---|
| 108 | 5.0 Incremental Development | 85 days? | Mon 25/6/2 | Mon 25/9/15 |
| 109 | 5.1 First Increment | 28 days | Mon 25/6/2 | Mon 25/7/7 |
| 110 | 5.1.1 Planning and Analysis | 12 days | Mon 25/6/2 | Tue 25/6/17 |
| 113 | 5.1.2 Design | 8 days | Wed 25/6/18 | Fri 25/6/27 |
| 114 | 5.1.2.1 Design login account and register new user screens | 4 days | Wed 25/6/18 | Mon 25/6/23 |
| 115 | 5.1.2.2 Design basic navigation flow after login | 3 days | Wed 25/6/25 | Fri 25/6/27 |
| 116 | 5.1.3 Implementation | 4 days | Mon 25/6/30 | Thu 25/7/3 |
| 117 | 5.1.3.1 Build registration UI | 1 day | Mon 25/6/30 | Mon 25/6/30 |
| 118 | 5.1.3.2 Build login UI | 2 days | Mon 25/6/30 | Tue 25/7/1 |
| 119 | 5.1.3.3 Implement user account creation with Firebase | 3 days | Tue 25/7/1 | Thu 25/7/3 |
| 120 | 5.1.3.4 Implement login functionality with Firebase Authentication | 3 days | Tue 25/7/1 | Thu 25/7/3 |
| 121 | 5.1.3.5 Integrate session handling for logged-in users | 3 days | Tue 25/7/1 | Thu 25/7/3 |
| 122 | 5.1.4 Testing | 3 days | Thu 25/7/3 | Sat 25/7/5 |
| 123 | 5.1.4.1 Test account registration with valid/invalid inputs | 1 day | Thu 25/7/3 | Thu 25/7/3 |
| 124 | 5.1.4.2 Test login with correct and incorrect credentials | 2 days | Fri 25/7/4 | Sat 25/7/5 |
| 125 | 5.1.4.3 Test access to system features after successful login | 3 days | Thu 25/7/3 | Sat 25/7/5 |
| 126 | 5.1.4.5 Test logout functionality | 3 days | Thu 25/7/3 | Sat 25/7/5 |

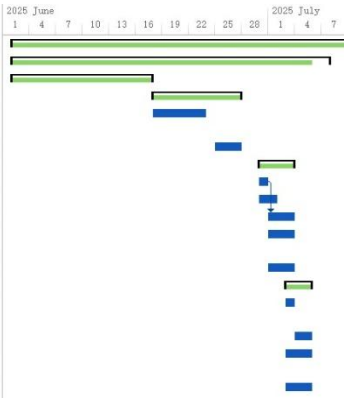



Figure 3. 10: Gantt Chart for First Increment.

### 3.5.6.2 Second Increment

| ID | Task Name | Duration | Start | Finish |
|---|---|---|---|---|
| 127 | 5.2 Second Increment | 27 days | Mon 25/7/7 | Tue 25/8/5 |
| 128 | 5.2.1 Planning and Analysis | 4 days | Mon 25/7/7 | Thu 25/7/10 |
| 129 | 5.2.1.1 Analyze requirements for inventory listing, filtering, searching | 1 day | Mon 25/7/7 | Mon 25/7/7 |
| 130 | 5.2.1.2 Define inventory item data structure | 1 day | Tue 25/7/8 | Tue 25/7/8 |
| 131 | 5.2.1.3 Plan database structure for inventory management | 1 day | Tue 25/7/8 | Tue 25/7/8 |
| 132 | 5.2.1.4 Analyze barcode scanning requirements | 1 day | Wed 25/7/9 | Wed 25/7/9 |
| 133 | 5.2.1.5 Define stock update process via scanning | 1 day | Wed 25/7/9 | Wed 25/7/9 |
| 134 | 5.2.1.6 Plan alert conditions | 1 day | Thu 25/7/10 | Thu 25/7/10 |
| 135 | 5.2.2 Design | 5 days | Fri 25/7/11 | Tue 25/7/15 |
| 136 | 5.2.2.1 Design screens | 5 days | Fri 25/7/11 | Tue 25/7/15 |
| 137 | 5.2.2.1.1 Inventory list screen | 1 day | Fri 25/7/11 | Fri 25/7/11 |
| 138 | 5.2.2.1.2 Add and Edit inventory item screens | 1 day | Sat 25/7/12 | Sat 25/7/12 |
| 139 | 5.2.2.1.3 Barcode scanner interface | 1 day | Sun 25/7/13 | Sun 25/7/13 |
| 140 | 5.2.2.1.4 Alerts and Notifications popups | 1 day | Mon 25/7/14 | Mon 25/7/14 |
| 141 | 5.2.2.1.5 Inventory Summary Report screen. | 1 day | Tue 25/7/15 | Tue 25/7/15 |
| 142 | 5.2.3 Implementation | 16 days | Wed 25/7/16 | Sat 25/8/2 |
| 143 | 5.2.3.1 Implement inventory item listing | 1 day | Wed 25/7/16 | Wed 25/7/16 |
| 144 | 5.2.3.2 Implement filtering and searching features | 1 day | Wed 25/7/16 | Wed 25/7/16 |
| 145 | 5.2.3.3 Implement add new item feature | 1 day | Thu 25/7/17 | Thu 25/7/17 |
| 146 | 5.2.3.4 Implement update item details feature | 1 day | Thu 25/7/17 | Thu 25/7/17 |
| 147 | 5.2.3.5 Integrate barcode scanning feature using device camera | 8 days | Fri 25/7/18 | Fri 25/7/25 |
| 148 | 5.2.3.6 Implement stock quantity updates after scann | 8 days | Fri 25/7/18 | Fri 25/7/25 |
| 149 | 5.2.3.7 Implement delete Item feature | 1 day | Mon 25/7/28 | Mon 25/7/28 |
| 150 | 5.2.3.8 Implement low stock alert notifications | 1 day | Tue 25/7/29 | Tue 25/7/29 |
| 151 | 5.2.3.9 Implement unmatched barcode alert | 2 days | Wed 25/7/30 | Thu 25/7/31 |
| 152 | 5.2.3.10 Implement inventory summary report generation | 1 day | Fri 25/8/1 | Sat 25/8/2 |
| 153 | 5.2.4 Testing | 3 days | Sun 25/8/3 | Tue 25/8/5 |
| 154 | 5.2.4.1 Test inventory list viewing, filtering, and searching | 1 day | Sun 25/8/3 | Sun 25/8/3 |
| 155 | 5.2.4.2 Test barcode scanning functionality | 2 days | Mon 25/8/4 | Tue 25/8/5 |
| 156 | 5.2.4.3 Test adding, updating, and deleting items | 2 days | Mon 25/8/4 | Tue 25/8/5 |
| 157 | 5.2.4.4 Test stock quantity updates via scanning | 2 days | Mon 25/8/4 | Tue 25/8/5 |
| 158 | 5.2.4.5 Test low stock alert triggering | 1 day | Tue 25/8/5 | Tue 25/8/5 |
| 159 | 5.2.4.6 Test summary report generation | 1 day | Tue 25/8/5 | Tue 25/8/5 |

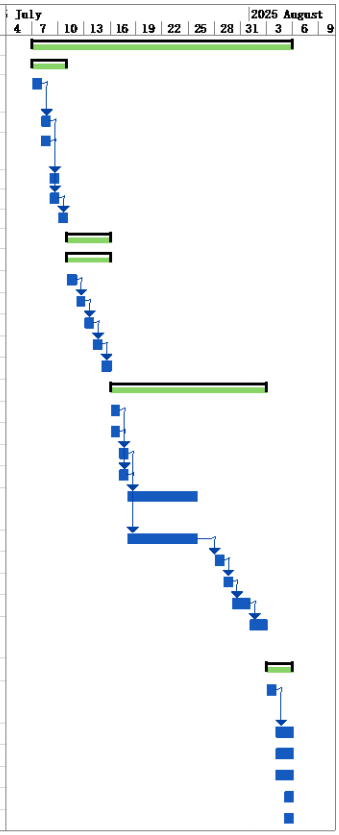



Figure 3. 11: Gantt Chart for Second Increment.

### 3.5.6.3 Third Increment

| ID | Task Name | Duration | Start | Finish |
|---|---|---|---|---|
| 160 | **5.3 Third Increment** | **28 days** | **Wed 25/8/6** | **Wed 25/9/10** |
| 161 | **5.3.1 Planning and Analysis** | **5 days** | **Wed 25/8/6** | **Mon 25/8/11** |
| 162 | 5.3.1.1 Define cargo and container attributes | 1 day | Wed 25/8/6 | Wed 25/8/6 |
| 163 | 5.3.1.2 Plan database structure for load plans | 1 day | Wed 25/8/6 | Wed 25/8/6 |
| 164 | 5.3.1.3 Define checklist workflow and data structure | 1 day | Thu 25/8/7 | Thu 25/8/7 |
| 165 | 5.3.1.4 Define load plan update mechanism | 3 days | Fri 25/8/8 | Mon 25/8/11 |
| 166 | 5.3.1.5 Plan the generation of printable PDF reports | 1 day | Fri 25/8/8 | Fri 25/8/8 |
| 167 | 5.3.1.6 Analyze QR code generation | 1 day | Sat 25/8/9 | Sat 25/8/9 |
| 168 | **5.3.2 Design screen** | **4 days** | **Tue 25/8/12** | **Fri 25/8/15** |
| 169 | 5.3.2.1 Load plan creation screen | 2 days | Tue 25/8/12 | Wed 25/8/13 |
| 170 | 5.3.2.2 Add custom cargo and container screen | 1 day | Wed 25/8/13 | Wed 25/8/13 |
| 171 | 5.3.2.3 Load plan editing screen | 1 day | Thu 25/8/14 | Thu 25/8/14 |
| 172 | 5.3.2.4 PDF layout for printable load plan report | 1 day | Thu 25/8/14 | Thu 25/8/14 |
| 173 | 5.3.2.5 Load plan history screen | 1 day | Fri 25/8/15 | Fri 25/8/15 |
| 174 | 5.3.2.6 Checklist screen | 1 day | Fri 25/8/15 | Fri 25/8/15 |
| 175 | **5.3.3 Implementation** | **15 days** | **Mon 25/8/18** | **Fri 25/9/5** |
| 176 | 5.3.3.1 Implement load plan generation | 8 days | Mon 25/8/18 | Wed 25/8/27 |
| 177 | 5.3.3.2 Implement add custom containers and cargo size feature | 1 day | Mon 25/8/18 | Mon 25/8/18 |
| 178 | 5.3.3.3 Implement drag-and-drop feature | 2 days | Thu 25/8/28 | Fri 25/8/29 |
| 179 | 5.3.3.4 Generate printable PDF reports | 1 day | Mon 25/9/1 | Mon 25/9/1 |
| 180 | 5.3.3.5 Implement QR code scanning | 2 days | Tue 25/9/2 | Wed 25/9/3 |
| 181 | 5.3.3.6 Implement checklist functionality | 2 days | Thu 25/9/4 | Fri 25/9/5 |
| 182 | **5.3.4 Testing** | **4 days** | **Sat 25/9/6** | **Wed 25/9/10** |
| 183 | 5.3.4.1 Test load plan creation | 1 day | Sat 25/9/6 | Sat 25/9/6 |
| 184 | 5.3.4.2 Test drag-and-drop updating of load plans | 1 day | Mon 25/9/8 | Mon 25/9/8 |
| 185 | 5.3.4.3 Test scanning QR codes to view correct load plan history | 1 day | Tue 25/9/9 | Tue 25/9/9 |
| 186 | 5.3.4.4 Test checklist functionality for checking off cargo items | 1 day | Wed 25/9/10 | Wed 25/9/10 |

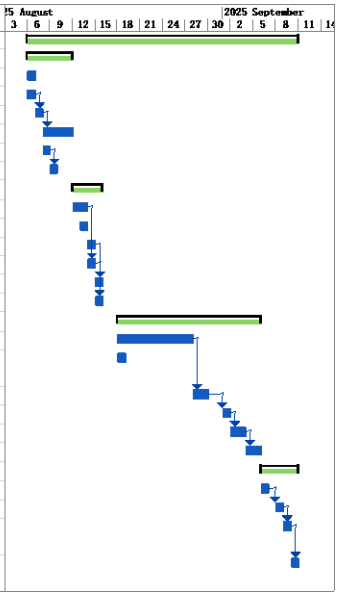



Figure 3. 12: Gantt Chart for Third Increment.

# CHAPTER 4

# PROJECT INITIAL SPECIFICATION

## 4.1 Introduction

This chapter includes detailed specifications for the inventory tracking and load planning mobile application being developed as part of this project. The following session outlines the methodology used to gather requirements, including fact-finding techniques such as interviewing experienced users. The system's functional and non-functional requirements will be defined, followed by use case modelling to describe the main interactions between the users and the system. Additionally, Interface Flow Diagram and Prototype Interface are also included to visually represent the app's structure, flow, and user interface.

## 4.2 Fact Finding

To gather relevant information for the development of an inventory tracking application and to verify the accuracy of the literature search information, interviews were conducted.

### 4.2.1 Interview

The first interview was conducted on 23 April 2025 at Ametal Tech Sdn Bhd and took about an hour. The interviewee invited was an experienced user of the relevant software. Ms Jesther, who is the Assistant Manager of the Logistics Department, has nearly 15 years of experience in logistics. The interviewees were selected based on her direct involvement with the load plan during her working life for so many years and could provide expert insights into my load planning functionality.

The interview adopted a semi-structured interview method, which combined pre-prepared questions with open-ended discussion. The interview began with a self-introduction and explained that the purpose of the interview was to collect users' feedback on the prototype and the functional requirements or expectations for the new application. The interview found that the load plan is currently managed using manual methods such as Microsoft Excel. Once the container size and the list of cargo to be loaded are finalized, the staff begin by

drawing the container dimensions to scale within Excel. They then proceed to manually draw each cargo item, which can number 20 or more, and label each one with its corresponding ID and name. After that, they attempt to arrange the cargoes within the container space, adjusting placements until the container is fully loaded. This manual process is time-consuming and labor-intensive. It usually takes at least an hour to complete a load plan, and even at least 3 to 4 loading plans need to be drawn up every day. This results in approximately four to five hours spent solely on load planning, effectively consuming nearly half of the working day.

The interviewee also highlighted key challenges faced during the current load planning process, including a high error rate and the need to repeatedly redo work due to mistakes or inefficient cargo arrangements. Using Excel to manually draw out each cargo item one by one is not only time-consuming but also prone to errors, especially dealing with a large number of goods with varying sizes and dimensions. After drawing the items, staff must manually try to fit them into the container, adjusting positions through trial and error to ensure all items fit properly. This lack of automation means that any change such as adjusting the order of loading, replacing an item, or modifying the container size requires the entire layout to be redone, often from scratch.

The interviewee also mentioned that she had previously tested a few existing load planning applications using trial versions. However, she found them not worth the investment, as her team only needed a single core function, which is basic load planning. Most of the available applications were designed with a wide range of advanced features, catering mainly to logistics or transport companies with more complex operational needs. In her case, the company only required a simple and focused solution. The existing applications she tested offered a wide range of features, such as route optimization, fleet tracking and multi-container management. However, only about 20% of these features, specifically the basic load planning function, were relevant to their needs. As a result, investing in full-featured commercial software was considered inefficient and unnecessary, especially for a business that does not operate as a full logistics or transport company.

The interviewee also mentioned the app should provide reporting and analysis features. For example, it would be possible to see how many items were shipped in a week and how many vehicles were used. This data would be very useful for her company's sales department because it would help evaluate transportation efficiency and better understand resource usage. For example, they could use this data to predict how many lorries would be needed to transport similar quantities of goods in the future.

After the interview, informal testing was carried out and the interviewee was given the opportunity to test prototypes that included the main features of the application so that the author could directly observe the user reactions and identify areas for improvement before moving into structured development. She was asked to complete some basic tasks such as generating a load plan using container sizes and cargoes provided in the software. While experiencing the prototype, she made some suggestions about the user interface. For example, she pointed out that the color-coded cargo blocks appeared visually appealing within the mobile application. However, it becomes ineffective when exported or printed as a PDF, because companies often print documents in black and white rather than in colour to save costs. She recommended using text-based labels directly on the cargo blocks instead, to ensure that printed versions remain readable.

### 4.2.2 Observation

The observation was carried out to gain a clearer understanding of the current workflow and challenges involved in inventory management, specifically how items are scanned and recorded when they are checked in and out of inventory. It was conducted on-site, where staff were observed performing their usual daily inventory tasks without any interference, to ensure the process remained authentic and uninterrupted.

During the observation, staff used a handheld barcode scanner to scan items one by one. After scanning, they manually updated the quantity of items that needed to be taken out. One key issue observed was related to the quantity input field in their existing system. It comes with a default value of “1,” which often causes confusion. In some cases, staff may forget to update the quantity, but the quantity field has a default value of 1, and the system cannot justify

whether this is the default value or the actual quantity to be taken out. This increases the risk of incorrect inventory records. The way to improve the system is to leave the quantity field blank by default. This way, if the staff forget to enter the correct quantity, the system can easily detect the missing input and prompt a reminder, thus helping reduce errors and improving accuracy in the process.

### 4.2.3 Summary for Interview and Observation

Table 4. 1: Summary for Interview and Observation

| Aspect | Interview | Observation |
|---|---|---|
| **Date and Location** | 23 April 2025, Ametal Tech Sdn Bhd | On-site at Ametal Tech Sdn Bhd |
| **Purpose** | Collect user feedback on load planning prototype and functional requirements | Understand workflow and challenges in inventory management, focusing on item scanning |
| **Method** | Semi-structured interview with pre-prepared questions and open-ended discussion | Non-intrusive observation of staff performing daily inventory tasks |
| **Key Participant** | Ms. Jesther, Assistant Manager of Logistics Department, 15 years of experience | Staff performing inventory tasks |
| **Current Tool** | Microsoft Excel | Handheld barcode scanner |
| **Current Process** | Draw container and cargo items to scale in Excel, followed by trial-and-error arrangement | Items scanned one-by-one using handheld barcode scanners, with manual quantity updates in the system |
| **Time Taken** | ~1 hour per plan;<br>3–4 plans daily<br>= ~4–5 hours per day | Not specified |

| | | |
|---|---|---|
| **Challenges** | - High error rate due to manual drawing and arrangement<br>- Time-consuming trial-and-error process<br>- Entire layout needs redone for any change<br>- Existing load planning apps too complex, with only 20% of features relevant | - Default quantity field value of "1" leads to errors if staff forget to update quantity |
| **User Feedback /Suggestions** | - Need for simple, focused load planning app<br>- Avoid feature bloat<br>- Reporting and analysis features | - Blank quantity field by default<br>- System prompts for missing quantity inputs to improve accuracy |
| **Prototype Feedback** | - Liked color-coded cargo blocks in-app<br>- Recommended using text labels for print readability | - N/A |

## 4.3 Requirement Specification

This section outlines the functional and non-functional requirements derived from the interview and the review of existing application. The functional requirement is divided into 3 main modules: User Management, Inventory Tracking and Load Planning.

### 4.3.1 Functional Requirements

**1. User Management**

Table 4. 2: Functional Requirement of User Management Module.

| | |
|---|---|
| FR01 | The system shall allow user to register a new user account. |
| FR02 | The system shall allow users to login via email and password. |

## 2. Inventory Tracking via barcode scanning

Table 4. 3: Functional Requirement of Inventory Tracking Module.

| | |
|---|---|
| FR03 | The system shall allow users to view a list of all inventory items. |
| FR04 | The system shall allow users to filter the inventory list by stock status and category. |
| FR05 | The system shall allow users to search inventory items using any keyword that matching across all item attributes. |
| FR06 | The system shall allow users to add a new inventory item. |
| FR07 | The system shall allow users to update the details of existing item. |
| FR08 | The system shall allow users to scan barcodes using the device camera |
| FR09 | The system shall allow users to update the stock quantity after scanning items in or out. |
| FR10 | The system shall allow users to delete an inventory item. |
| FR11 | The system shall visually indicate when the stock quantity is lower than the specified stock level. |
| FR12 | The system shall allow user to receive alerts if a scanned barcode does not match any existing inventory item. |

## 3. Load Planning

Table 4. 4: Functional Requirement of Load Planning Module.

| | |
|---|---|
| FR13 | The system shall allow user to generate a load plan based on the selected cargo and container. |
| FR14 | The system shall allow user to add custom containers and cargo size if the desired size is not available in the predefined list. |
| FR15 | The system shall allow user to update the load plan by drag and drop cargo items. |
| FR16 | The system shall allow user to export the load plan into printable PDF report. |
| FR17 | The system shall allow user to view load plan history via scanning the QR code from the PDF load plan. |

| | |
|---|---|
| FR18 | The system shall allow user to view the checklist for double-checking the arrangement status of each cargo item. |
| FR19 | The system shall allow user to mark items as checked off in the checklist after they have been successfully arranged in the load plan. |

### 4.3.2 Non-functional requirements

Table 4. 5: Non-Functional Requirement.

| | | |
|---|---|---|
| NFR01 | The system shall respond to user actions, such as barcode scans and item updates, in under 2 seconds. | Performance |
| NFR02 | The system shall have an intuitive, easy-to-navigate interface that users can quickly understand. | Usability |
| NFR03 | The system shall be compatible with a wide range of Android devices and versions. | Portability |
| NFR04 | The system shall remain accessible to users at any time, as long as they have an internet connection. | Availability |
| NFR05 | The system shall be able to access and use the mobile device's camera. | Compatibility |
| NFR06 | The system shall ensure input validation and display appropriate error messages for invalid inputs. | Usability |

## 4.4 Use Case Modelling

To enhance clarity and avoid overcrowding, the use case diagram is split into two separate diagrams. The first use case diagram combines the User Management and Inventory Tracking features, while the second use case diagram focuses on Load Planning.

### 4.4.1 Use Case Diagram

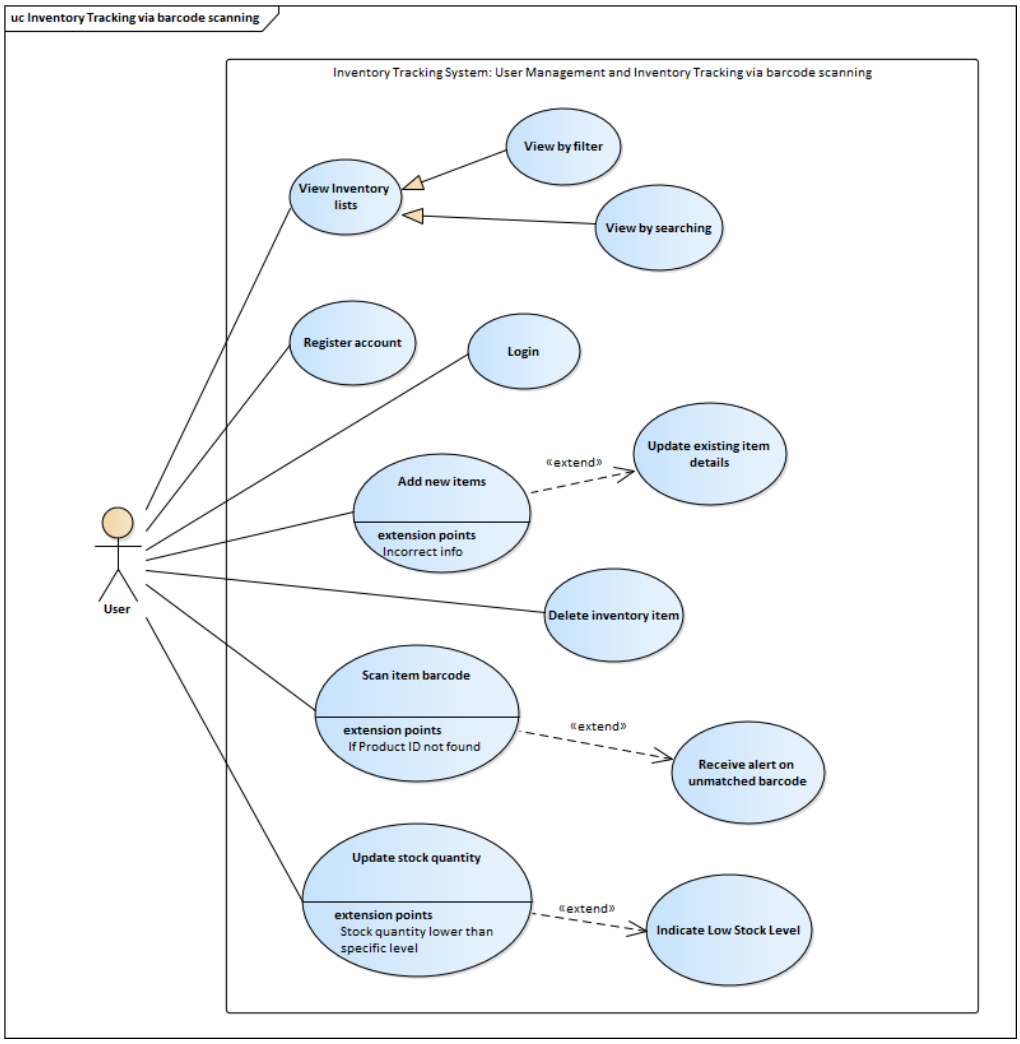



Figure 4. 1: Use case Diagram of Inventory Tracking System: User Management and Inventory Tracking via barcode scanning.

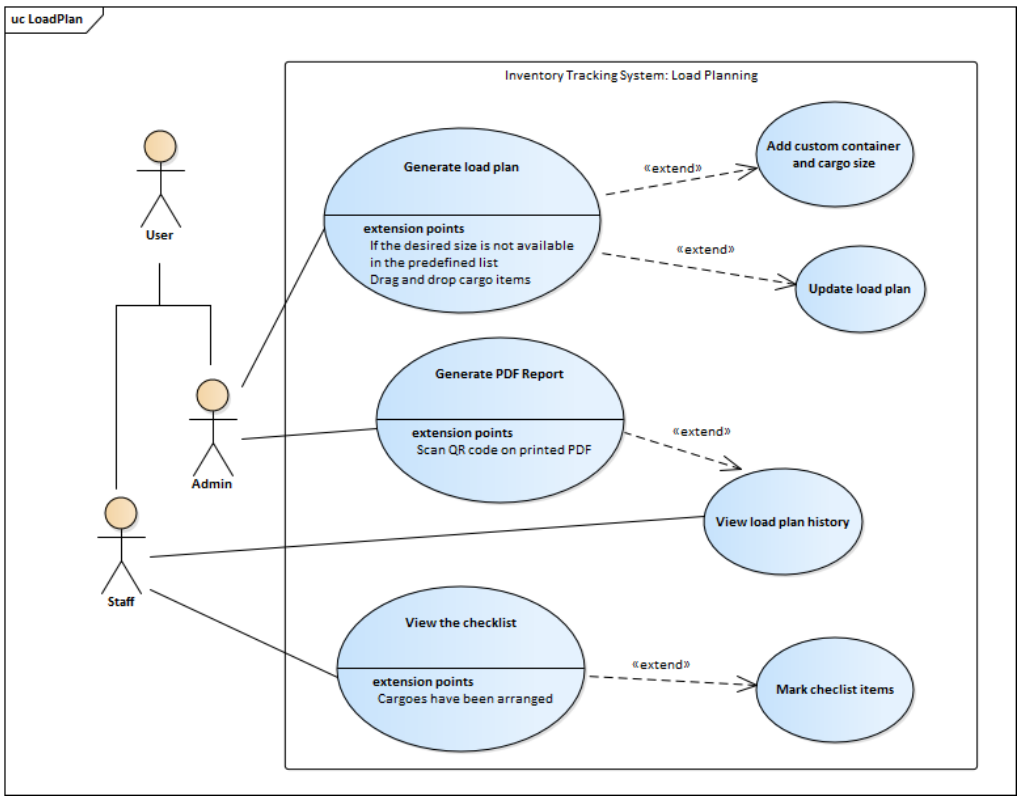



Figure 4. 2: Use Case Diagram of Inventory Tracking System: Load Planning.

### 4.4.2 Use Case Description

**Table 1: Use Case Description of Register account**

Table 4. 6: Use Case Description of Register account.

| | | |
|---|---|---|
| Use Case Name: Register account | ID: UC01 | Importance Level: High |
| Primary Actor: User | Use Case Type: Detail, Real | |
| Stakeholders and Interests: User – Wants to create new user accounts | | |
| Brief Description: This use case allows a user to create a new user account using a valid email and password. | | |
| Trigger: User wants to create a new user account for a new user. | | |
| Relationships:<br>Association : User<br>Include : | | |

| |
|---|
| Extend : Login<br>Generalization: N/A |
| Normal Flow of Events:<br>1. The user clicks the '+' button on the User Management screen.<br>2. The system displays the add new user modal.<br>3. The user inputs required information such as email, name and password.<br>4. The user clicks the 'Add' button to submit the form.<br>5. The system validates the email format. Continue to Sub-flows 5.1 or 5.2.<br>6. The system checks whether the email is already registered. Continue to Sub-flows 6.1 or 6.2.<br>7. The system creates a new account.<br>8. The system redirects the user to the User Management screen. |
| Sub-flows:<br>5.1 If the email format correct:<br>5.1.1 Continue to Flow 6.<br>5.2 If the email format is incorrect:<br>5.2.1 The system displays an error message: "Invalid email address."<br>5.2.2 The user is prompted to correct the information and try again.<br>5.2.3 After correction, continue to Flow 6.<br><br>6.1 If the email is already registered:<br>6.1.1 The system displays an error message: "This email is already registered."<br>6.1.2 The user is prompted to try a different email.<br>6.2 If the email is not yet registered:<br>6.2.1 The system creates a new user account.<br>6.2.3 Continue to Flow 7. |
| Alternate/Exceptional Flows: |

Table 4. 7: Use Case Description of Login account.

| Use Case Name: Login account | ID: UC02 | Importance Level: High |
|---|---|---|
| Primary Actor: User | Use Case Type: Detail, Real | |
| Stakeholders and Interests: User – Wants to access the inventory management system using their assigned email and password. | | |
| Brief Description: This use case allows a user member to log in to the system using their assigned email and password. | | |
| Trigger: The user wants to log in to his/her account. | | |
| Relationships:<br>Association : User<br>Include : N/A<br>Extend : N/A<br>Generalization: N/A | | |
| Normal Flow of Events:<br>1. The user clicks the 'Login' button from the Welcome Screen.<br>2. The system displays the login form.<br>3. The user enters the assigned email and password.<br>4. The user clicks the 'Login' button to submit credentials.<br>5. The system validates the email format. Continue to Sub-flows 5.1 or 5.2.<br>6. The system checks the credentials against the authentication database. Continue to Sub-flows 6.1 or 6.2.<br>7. Upon successful login, the system redirects the user to the home screen. | | |
| Sub-flows:<br>5.1 If the email format is correct: | | |

| |
|---|
| 5.1.1 Continue to Flow 6.<br>5.2 If the email format is incorrect:<br>5.2.1 The system displays an error: “Please enter a valid email address.”<br>5.2.2 The user corrects the input and retries.<br>5.2.3 After correction, continue to Flow 6.<br><br>6.1 If the credentials are valid:<br>6.1.1 The system logs in to the user and redirects to the home screen.<br>6.2 If the credentials are invalid:<br>6.2.1 The system displays an error message: “Incorrect email or password.”<br>6.2.2 The user is prompted to retry or reset their password. |
| Alternate/Exceptional Flows: |

Table 4. 8: Use Case Description of Scan item barcode.

| | | |
|---|---|---|
| Use Case Name: Scan item barcode | ID: UC03 | Importance Level: High |
| Primary Actor: User | Use Case Type: Detail, Real | |
| Stakeholders and Interests: User – wants to scan barcodes using the device camera to quickly identify items and update stock when adding or removing them. | | |
| Brief Description: This use case allows staff to scan an item’s barcode using the device camera. The system reads the barcode, checks if the item exists in the inventory, and displays the relevant details. | | |
| Trigger: The user wants to update stock by scanning one or more items. | | |
| Relationships:<br>Association : User<br>Include : N/A | | |

| |
|---|
| Extend : Receive alert on unmatched barcode<br>Generalization: N/A |
| Normal Flow of Events:<br>1. The user clicks the “Scan” icon button on the navigation bar from the home screen.<br>2. The system opens the barcode scanner camera interface.<br>3. The user scans a product barcode using the device’s camera.<br>4. The system detects and reads the scanned value.<br>5. The system searches for a matching Product ID in the inventory database. Continue sub-flows 5.1 or 5.2. |
| Sub-flows:<br>5.1 If Product ID is found:<br>5.1.1 The system fetches and displays product image, ID, name, quantity and stock status.<br>5.1.2 The staff adjust the quantity of scanned items.<br>5.1.3 The staff proceed to scan the next item by clicking the ‘scan’ icon button.<br>5.2 If Product ID is not found:<br>5.2.1 The system alerts the user that the item is not found in the inventory.<br>5.2.2 The system returns the staff to the scan interface. |
| Alternate/Exceptional Flows:<br>A1 - Camera access not granted:<br>A1.1 The system displays: "Camera permission required to scan barcodes."<br>A1.2 The staff is prompted to enable camera permissions in the device settings. |

Table 4. 9: Use Case Description of Update stock quantity.

| Use Case Name: Update stock quantity | ID: UC04 | Importance Level: High |
|---|---|---|
| Primary Actor: User | Use Case Type: Detail, Real | |

| | |
|---|---|
| Stakeholders and Interests: User – Wants to update the stock quantity for one or multiple items. | |
| Brief Description: This use case allows user to update the stock quantity of items in two different ways. For individual updates, staff can go to the product detail screen and manually add or reduce the quantity. For handling multiple items efficiently, they can scan barcodes one by one and update the stock immediately after each scan. | |
| Trigger: The user wants to update the stock quantity. | |
| Relationships:<br>Association : User<br>Include : N/A<br>Extend : Indicate Low Stock Level<br>Generalization: N/A | |
| Normal Flow of Events:<br>1. The user wants to update the stock quantity of one or more items.<br>2. The user proceeds with one of the methods. Continue sub-flows 2.1 or 2.2. | |
| Sub-flows:<br>2.1 Manual update from product description screen:<br>2.1.1 The user opens the product's description screen.<br>2.1.2 They tap the “IN” or “OUT” button.<br>2.1.3 The system prompts for a quantity input.<br>2.1.4 The user enters the desired quantity and confirms.<br>2.1.5 The system validates and updates the quantity.<br>2.1.6 The system checks against the low stock threshold. Continue sub-flow 3.1 or 3.2.<br>2.2 Update via barcode scanning: | |

| |
|---|
| 2.2.1 The user clicks the ‘Scan’ icon button on the home screen.<br>2.2.2 The system opens the barcode scanner interface.<br>2.2.3 The user scans an item’s barcode.<br>2.2.4 Upon a successful scan, the system shows the item details.<br>2.2.5 The user adjusts the quantity.<br>2.2.6 If there are more items to scan, the user clicks the ‘Scan’ icon again and repeats from Step 2.2.3. otherwise,<br>2.2.7 If there are no more items, the user chooses either ‘Product In’ or ‘Product Out’.<br>2.2.8 The system updates the stock accordingly.<br>2.2.9 The system checks against the low stock threshold. Continue sub-flow 3.1 or 3.2.<br><br>3.1 If quantity is above specific level:<br>3.1.1 The system confirms the update and returns to the previous screen.<br>3.2 If quantity is below specific level:<br>3.2.1 The system highlights the item with a low stock visual indicator using a red dot.<br>3.2.2 Use case ends. |
| Alternate/Exceptional Flows:<br>A1. Quantity set to 0 or negative:<br>A1.1 System displays: “Invalid quantity. Must be at least 1.”<br>A1.2 User is prompted to adjust the quantity before proceeding. |

Table 4. 10: Use Case Description of View inventory list.

| | | |
|---|---|---|
| Use Case Name: View inventory list | ID: UC05 | Importance Level: High |
| Primary Actor: User | Use Case Type: Detail, Real | |
| Stakeholders and Interests: User – Want to view the complete inventory or locate specific items using filters or search functionality. | | |

Brief Description: This use case allows users to view the full list of current inventory items. Users can also narrow the list by applying filters or searching using keywords.

Trigger: The user wants to view all inventory items, filter the list, or search for specific products.

Relationships:

Association : User

Include : N/A

Extend : N/A

Generalization: View by filtering, View by searching

Normal Flow of Events:

1. The user navigates to the inventory list screen by clicking the 'Product' tab in the navigation bar on the home screen.
2. The system retrieves and displays all current inventory items from the database.
3. Each item is shown with its product image, ID, name and current stock quantity with units.
4. The user can optionally choose to either apply filters or perform a search using keywords. Continue sub-flows 4.1 or 4.2.

Sub-flows:

4.1 Filter by category or stock status

4.1.1 The user selects a product category or chooses a stock status such as In Stock, Low Stock or Out of Stock.

4.1.2 The system applies the filters and updates the list.

4.1.3 The use case ends.

4.2 Search using keywords

4.2.1 The user enters a keyword into the search bar.

4.2.2 The system displays all items matching the entered keyword.

4.1.3 The use case ends.

| Alternate/Exceptional Flows:<br>A1 No results from filter:<br>A1.1 System displays: “No matching items found.” |
|---|

Table 4. 11: Use Case Description of Add new items.

| Use Case Name: Add new items | ID: UC06 | Importance Level: High |
|---|---|---|
| Primary Actor: User | Use Case Type: Detail, Real | |
| Stakeholders and Interests: User – Wants to add new inventory items when receiving new products. | | |
| Brief Description: This use case allows user to manually register a new item by entering product details such as name, category, stock quantity, image and minimum stock quantity. If have incorrect or incomplete information, the user may also proceed to update the item details. | | |
| Trigger: The user wants to add a new product to the inventory database. | | |
| Relationships:<br>Association : User<br>Include : N/A<br>Extend : Update existing item details<br>Generalization: N/A | | |
| Normal Flow of Events:<br>1. The user clicks the “+” button on the inventory list screen.<br>2. The system displays a form to enter new item’s details.<br>3. The user fills in the required fields such as Product image, Product name, Current stock quantity, Category, and Minimum stock quantity.<br>4. The user clicks the “Save” button.<br>5. The system validates the entered information and saves the new item | | |

| |
|---|
| to the database.<br>6. A success message is displayed: “Item added successfully.”<br>7. The new item appears in the inventory list.<br>8. If the user notices a typo or incorrect data, continue to sub-flow 8.1. Otherwise, the use case ends. |
| Sub-flows:<br>8.1 Correcting incorrect info:<br>8.1.1 The user selects the newly added item from the inventory list.<br>8.1.2 The system displays the product description screen.<br>8.1.3 The user clicks the three dots (menu) icon at the top right corner and selects "Edit".<br>8.1.4 The system navigates to the edit item screen.<br>8.1.5 The user updates the necessary fields and clicks “Save.”<br>8.1.6 The system saves the changes and displays: “Item details updated successfully.”<br>8.1.7 The updated item appears in the inventory list. Sub-flow 8.1 ends. |
| Alternate/Exceptional Flows: |

Table 4. 12: Use Case Description of Delete inventory items.

| | | |
|---|---|---|
| Use Case Name: Delete inventory items | ID: UC07 | Importance Level: High |
| Primary Actor: User | Use Case Type: Detail, Real | |
| Stakeholders and Interests: User – Wants to delete an inventory item from the system when it is no longer needed. | | |
| Brief Description: This use case allows user to delete an existing inventory item from the system. | | |
| Trigger: User decides to remove an item from the inventory list. | | |

| |
|---|
| Relationships:<br>Association : User<br>Include : N/A<br>Extend : N/A<br>Generalization: N/A |
| Normal Flow of Events:<br>1. The user selects an item from the inventory list.<br>2. The system navigates to the product description screen.<br>3. The user clicks the 'three dots' icon at the top right corner and selects the 'trash bin' icon button.<br>4. The system displays a confirmation dialog: "Are you sure you want to delete this item?"<br>5. The user confirms the deletion by clicking "Delete".<br>6. The system deletes the item from the database.<br>7. A success message is displayed: "Item deleted successfully."<br>8. The system returns the user to the inventory list screen. |
| Sub-flows: |
| Alternate/Exceptional Flows: |

Table 4. 13: Use Case Description of Generate Load Plan.

| | | |
|---|---|---|
| Use Case Name: Generate Load Plan | ID: UC08 | Importance Level: High |
| Primary Actor: User | Use Case Type: Detail, Real | |
| Stakeholders and Interests: User – Wants to generate a load plan to arrange selected cargo items into a container | | |
| Brief Description: This use case allows the user to generate a load plan by arranging selected cargo items inside a container based on container size and cargo dimensions. It also allows the user to update the arrangement using drag-and-drop functionality as needed. | | |

| |
|---|
| Trigger: Users want to arrange selected items for shipment within a container. |
| Relationships:<br>Association : User<br>Include : N/A<br>Extend : Add custom container and cargo size, Update load plan<br>Generalization: N/A |
| Normal Flow of Events:<br>1. The user navigates to the Load Plan section from the home menu.<br>2. The system displays options which are Plan Load, Set Common Size, and History.<br>3. The user selects Plan Load to start creating a new load plan.<br>4. The user inputs or selects the container size. Continue with sub-flow 4.1 or 4.2.<br>5. The user inputs or selects the cargo dimensions and quantity. Continue with sub-flow 6.1 or 6.2.<br>6. The user confirms the container and cargo selections.<br>7. The system retrieves the container, and cargo details and automatically generates a load plan, arranging cargo inside the container.<br>8. The system displays confirmation details, including selected container, selected cargo items with quantities, an Actual Ratio Diagram showing the arrangement, Calculated used space, free space, and total container space.<br>9. After the load plan is generated, the user may optionally rearrange cargo items manually. Continue with Sub-flow 9.1 if needed.<br>10. The user reviews the arrangement and clicks the ‘Save’ button.<br>11. A confirmation message appears: “Saved successfully.” |
| Sub-flows: |

| |
|---|
| 4.1 If using a preset container size:<br>4.1.1 The user selects a container from the predefined list. Continue flow-no 5.<br>4.2 If adding a custom container size:<br>4.2.1 The user clicks the ‘Set common size’ button on the Load Plan tab.<br>4.2.2 The user enters the container’s length, width, height and maximum weight capacity.<br>4.2.3 The system validates and saves the custom size input. Continue flow-no 5.<br><br>5.1 If using preset cargo dimensions and quantity:<br>5.1.1 The users select the predefined cargo items.<br>5.1.2 The user adjusts the quantity. Continue flow-no 6.<br>5.2 If adding custom cargo size:<br>5.2.1 The iser enters the cargo dimensions and quantity. Continue flow-no 6.<br><br>9.1 Rearranging Cargo Items Manually:<br>9.1.1 The user selects a cargo item inside the 3D load plan.<br>9.1.2 The user drags and drops the cargo item to a new position within the container.<br>9.1.3 The system dynamically updates the load plan layout based on the new arrangement.<br>9.1.4 The user repeats the drag-and-drop action as needed to finalize the arrangement.<br>9.1.5 Once satisfied, continue flow no-10. |
| Alternate/Exceptional Flows: |

Table 4. 14: Use Case Description of View the checklist.

| | | |
|---|---|---|
| Use Case Name: View the checklist | ID: UC09 | Importance Level: High |

| Primary Actor: User | Use Case Type: Detail, Real |
|---|---|
| Stakeholders and Interests: User- wants to view a checklist that shows which cargo items still need to be arranged within the current load plan, and mark items as placed once arranged. | |
| Brief Description: This use case allows user to display a checklist of cargo items that not yet been arranged. | |
| Trigger: The user member wants to see which cargo items still require arrangement. | |
| Relationships:<br>Association : User<br>Include : N/A<br>Extend : Mark Checklist Item<br>Generalization: N/A | |
| Normal Flow of Events:<br>1. The user scans the barcode on the PDF load plan.<br>2. The system navigates to the corresponding load plan description page.<br>3. The user clicks the “Checklist” button.<br>4. The system retrieves the list of cargo items that need to be loaded into the container.<br>5. The system displays the checklist, showing details for each cargo item such as cargo name, quantity, and dimensions.<br>6. As the user physically arranges each cargo item into the container, they mark the corresponding checklist item.<br>7. The system marks the item as completed with a checkmark.<br>8. The user continues arranging and marking items until all cargo items are completed.<br>9. Once all items are marked as placed, the user clicks Save.<br>10. The system updates the load plan status from Pending to Finished. | |

| Sub-flows: |
|---|
| Alternate/Exceptional Flows: |

Table 4. 15: Use Case Description of Generate PDF report.

| Use Case Name: Generate PDF report | ID: UC10 | Importance Level: High |
|---|---|---|
| Primary Actor: User | Use Case Type: Detail, Real | |
| Stakeholders and Interests:<br>User<br>- want to generate a printable version of the finalized load plan that includes cargo arrangement, container details, and a QR code for quick future access.<br>-wants to retrieve and view the load plan details by scanning the QR code. | | |
| Brief Description: This use case allows user to generate a PDF report of a load plan which includes a QR code that links directly back to the system for easy future access. | | |
| Trigger: The user finalizes a load plan and chooses to export it as a PDF. | | |
| Relationships:<br>Association : User<br>Include : N/A<br>Extend : View Load Plan History<br>Generalization: N/A | | |
| Normal Flow of Events:<br>1. The user opens the load plan description page.<br>2. The user clicks the “Generate PDF” button.<br>3. The system compiles the load plan data such as Load plan ID, container details, total used space and remaining space, cargo arrangement order, and a QR code into a formatted PDF.<br>4. The system generates the PDF file and displays the message: | | |

| |
|---|
| "Generated Successfully!"<br>5. The user clicks the "View" button to preview the generated PDF.<br>6. The PDF file can be printed or distributed as needed. Proceed with sub-flow 6.1 or 6.2. |
| Sub-flows:<br>6.1 Staff views load plan history via scanning QR Code:<br>6.1.1 The user scans the QR code from the printed PDF copy.<br>6.1.2 The system retrieves and opens the associated load plan description page.<br>6.1.3 The user can view the load plan details, including cargo arrangement and container info.<br>6.1.4 The use case ends.<br>6.2 User does not scan the QR code:<br>6.2.1 If the QR code is not scanned, no action is triggered.<br>6.2.2 The use case ends. |
| Alternate/Exceptional Flows: |

## 4.5 Interface Flow Diagram

This section shows the interface flow of the application, illustrating how users navigate between different screens and interact with the system.

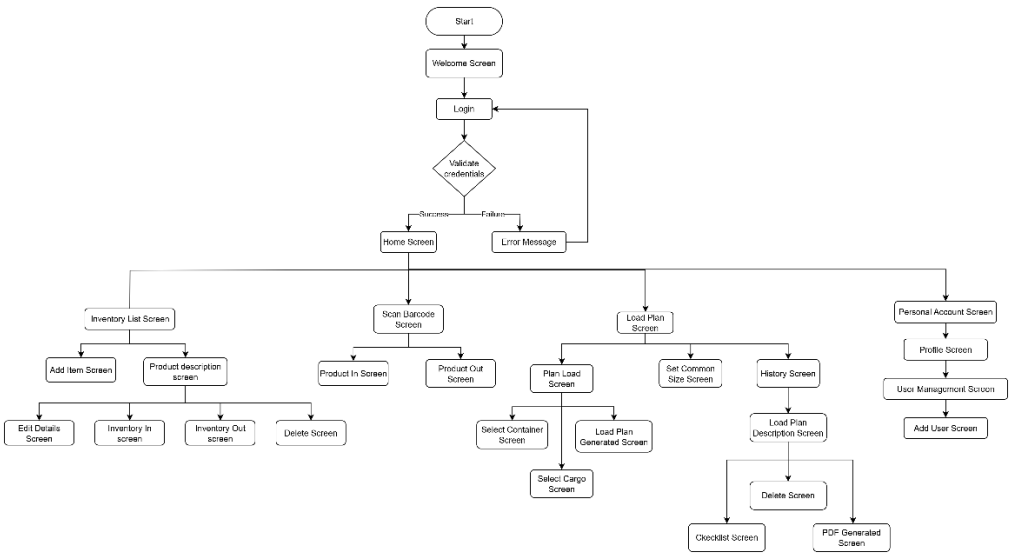



Figure 4. 3: Interface Flow Diagram of Proposed System.

### 4.5.1 User Management Module

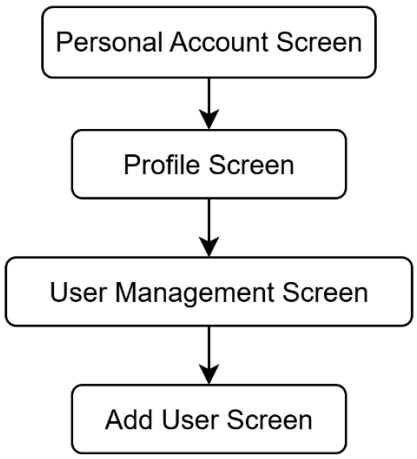



Figure 4. 4: Interface Flow in User Management Module.

### 4.5.2 Inventory Tracking via Barcode Scanning Module

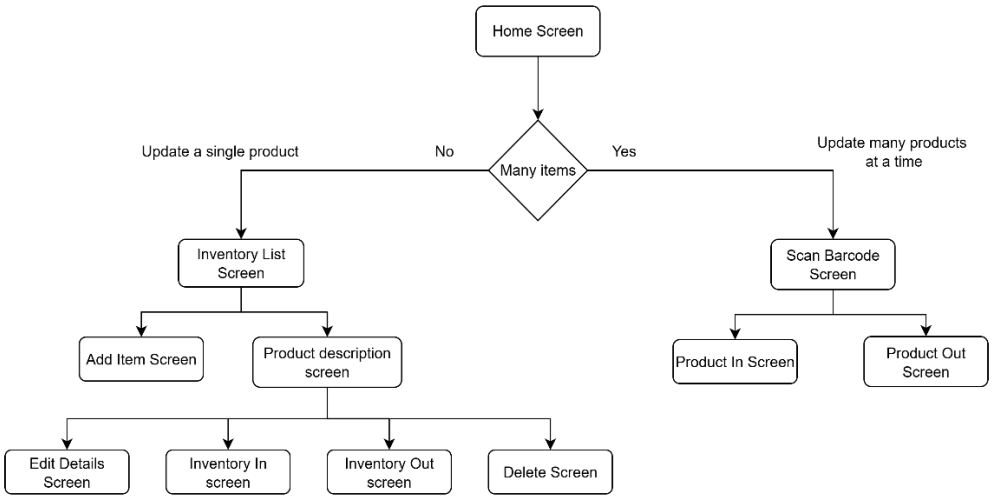



Figure 4. 5: Interface Flow in Inventory Tracking Module.

### 4.5.3 Cargo Load Planning Module

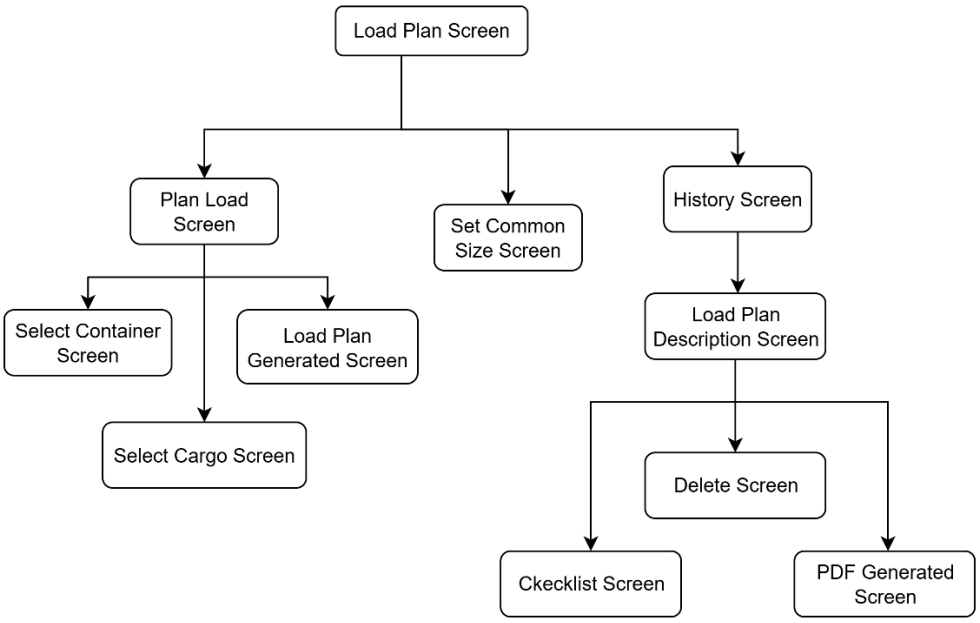



Figure 4. 6: Interface Flow in Cargo Load Planning Module.

## 4.6 Prototype Interface

The prototype development was developed based on the three main modules specified in the project scope, which will include the module of user management, inventory tracking via barcode scanning, and load planning.

### 4.6.1 User Management Module

The User Management Module enables users to create and access their own accounts through a secure registration and login system. It ensures that only registered users can log in to the application using their email and password. Once authenticated, users are granted access to all system features such as inventory tracking and load planning. Figure 4.4 illustrates the process of registering a new user account within the system.

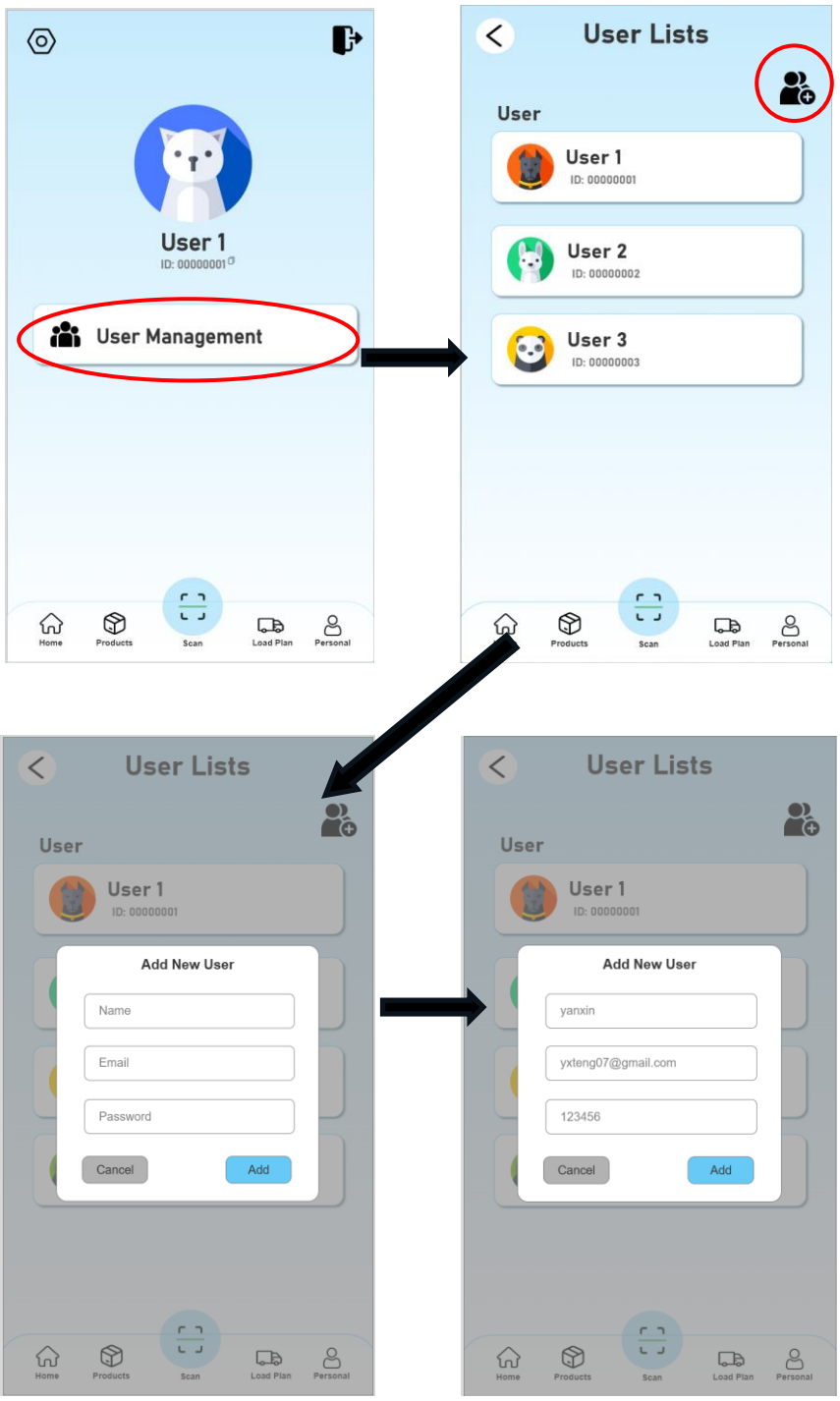

User 1
ID: 00000001
User Management
Home
Products
Scan
Load Plan
Personal
User Lists
User
User 1
ID: 00000001
User 2
ID: 00000002
User 3
ID: 00000003
Add New User
Name
Email
Password
Cancel
Add
yanxin
yxteng07@gmail.com
123456


Figure 4. 7: Register a New Account.

The Login feature allows users to access the system using the email and password with the credentials assigned to them. Figure 4.8 show that the welcomes screen and login form.

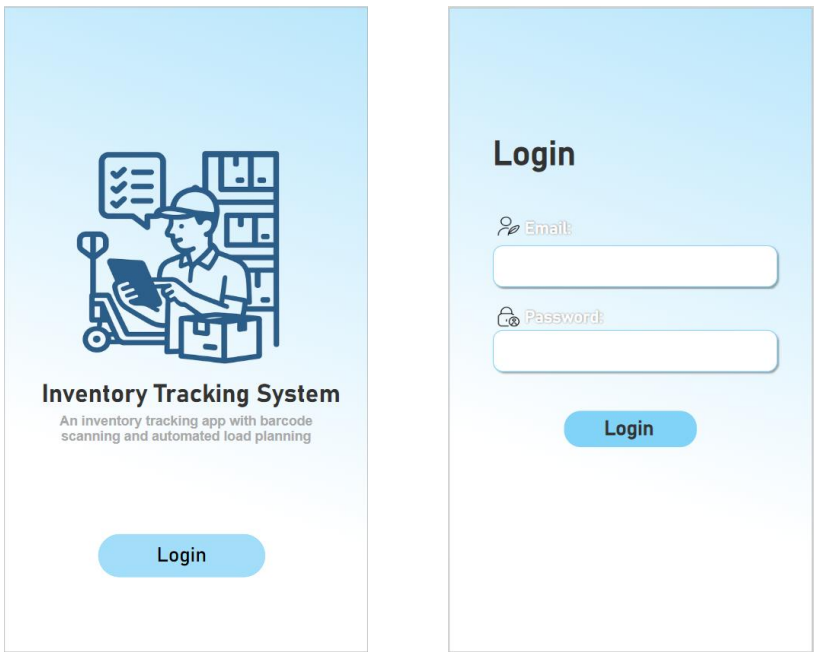



Figure 4. 8: Login feature by Staff.

### 4.6.2 Inventory Tracking via Barcode Scanning Module

It provides a complete inventory list with options to filter and search items by attributes such as category or quantity. Using the device camera, users can quickly scan barcodes to update stock quantities for check-in and check-out

activities. In addition, users can manually add, update, or delete inventory records as needed. The system also provides automatic alerts when stock levels fall below a predefined threshold or when a scanned barcode does not match any existing item, thereby helping to maintain accuracy and prevent errors in inventory management.

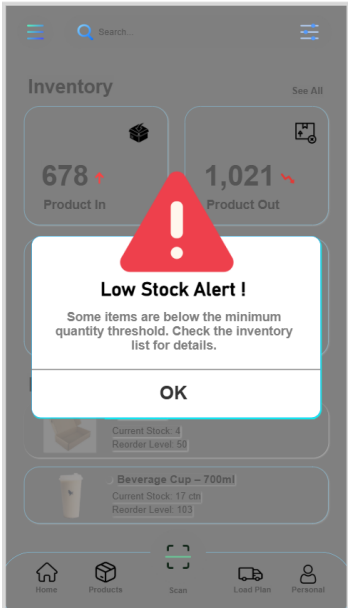



Figure 4. 9: Low Stock Alert Message Displayed After User Login.

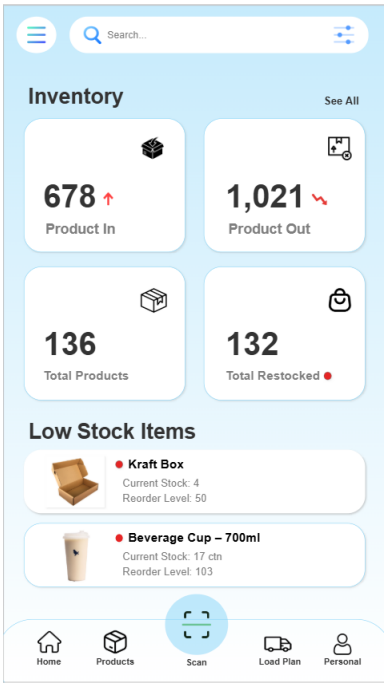



Figure 4. 10: Home Page.


Figure 4. 11: Inventory Items List Screen.


Figure 4. 12: Filter Feature by Category and Quantity.

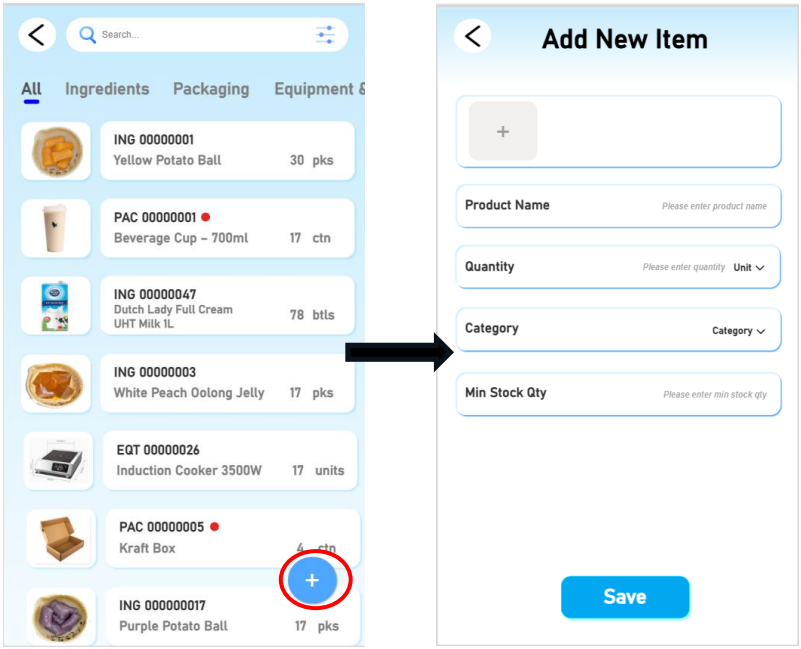



Figure 4. 13: Form Interface for Adding a New Inventory Item

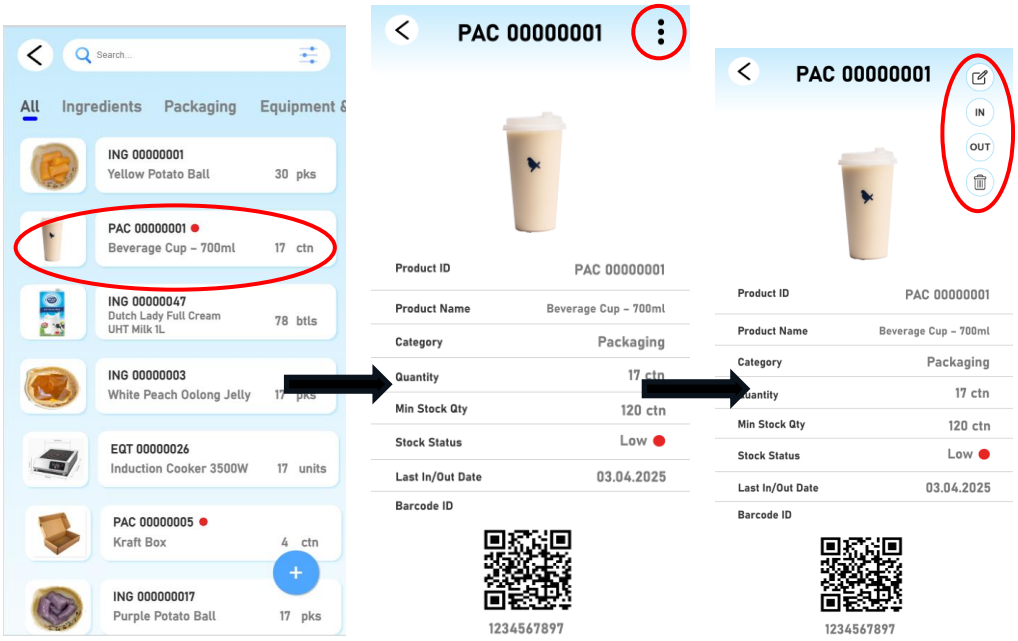



Figure 4. 14: Product Description Page.

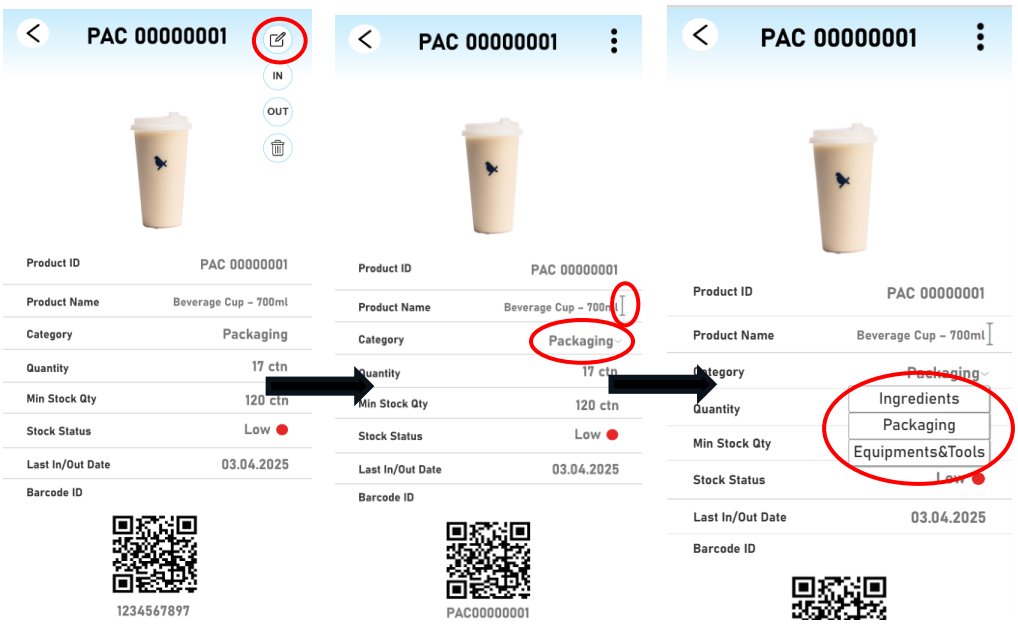



Figure 4. 15: Interface for Editing Inventory Item Details.

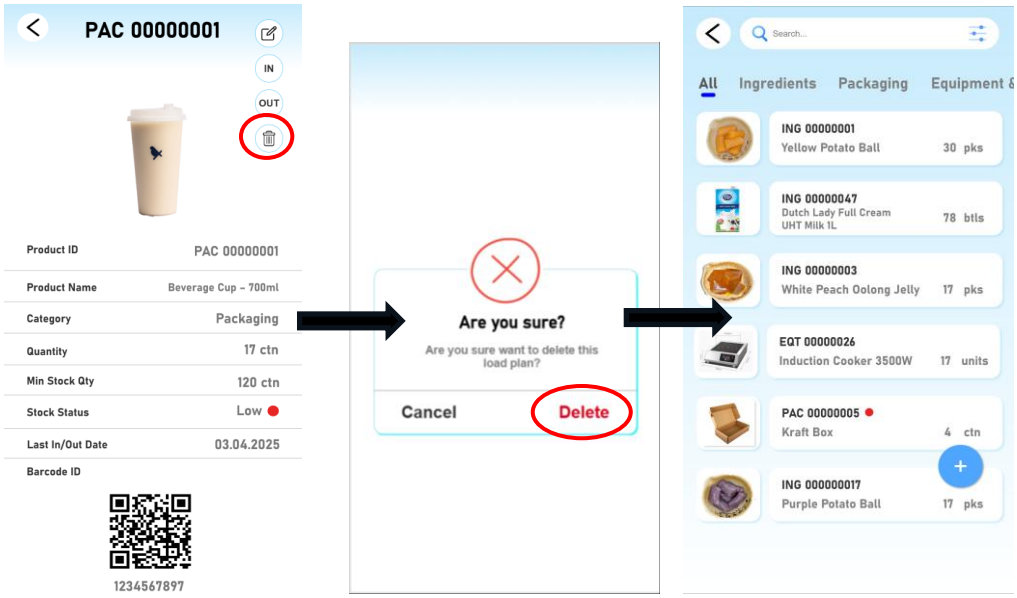



Figure 4. 16: Confirmation Message for Deleting an Inventory Item.

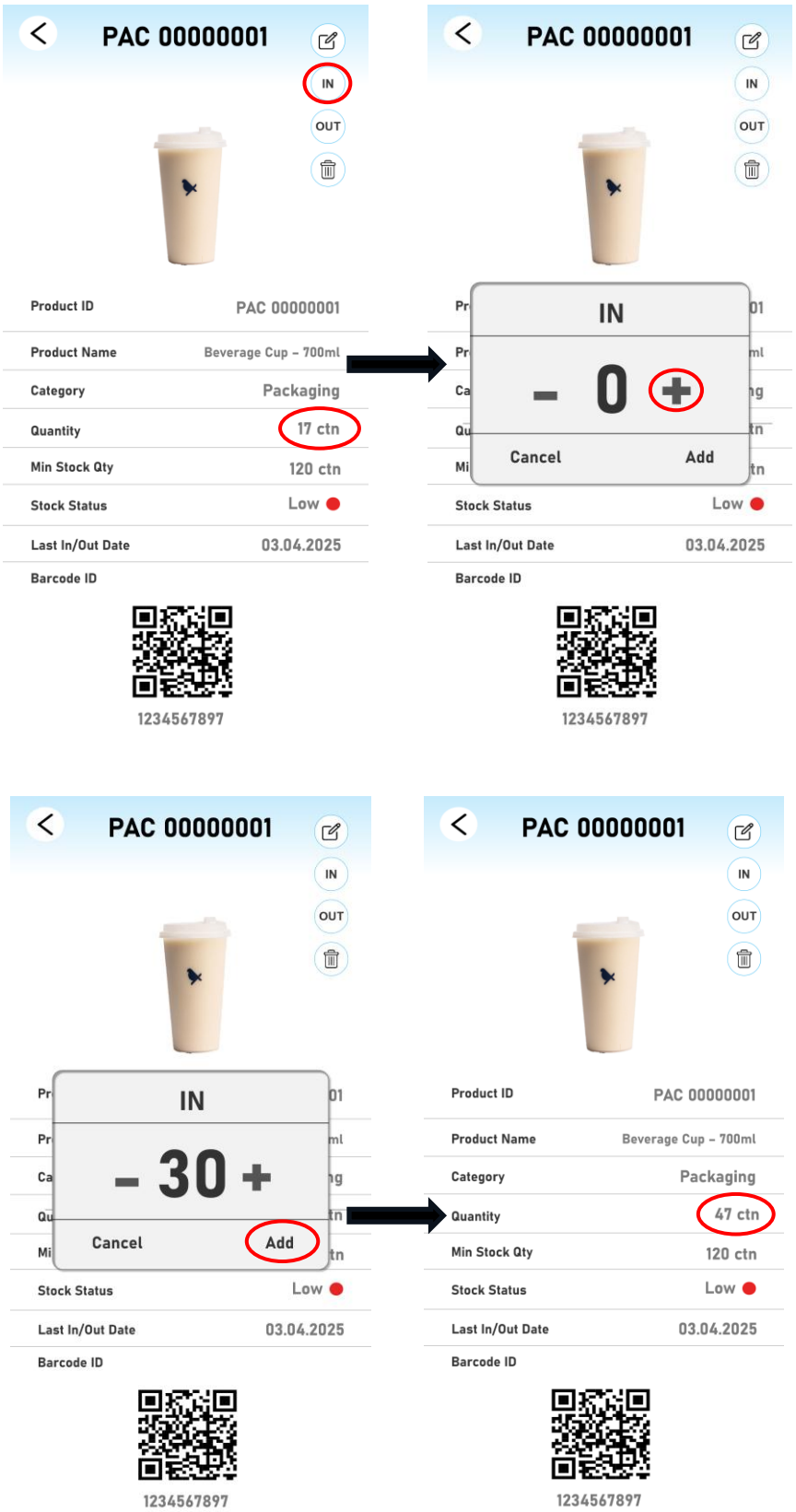



Figure 4. 17: Stock Update for a Single Inventory Item ('In' Button).

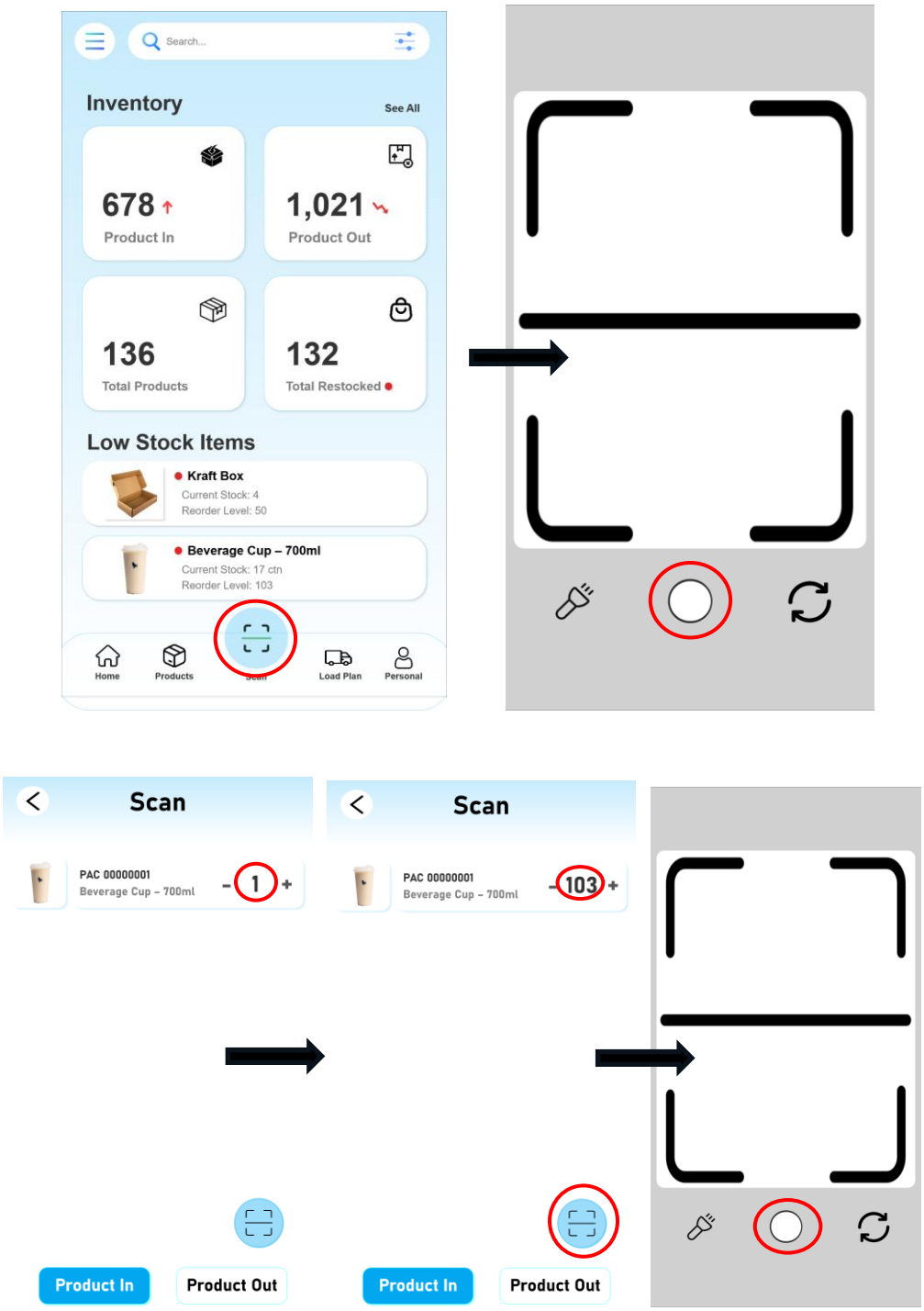



Figure 4. 18: Steps to Scan a Barcode.

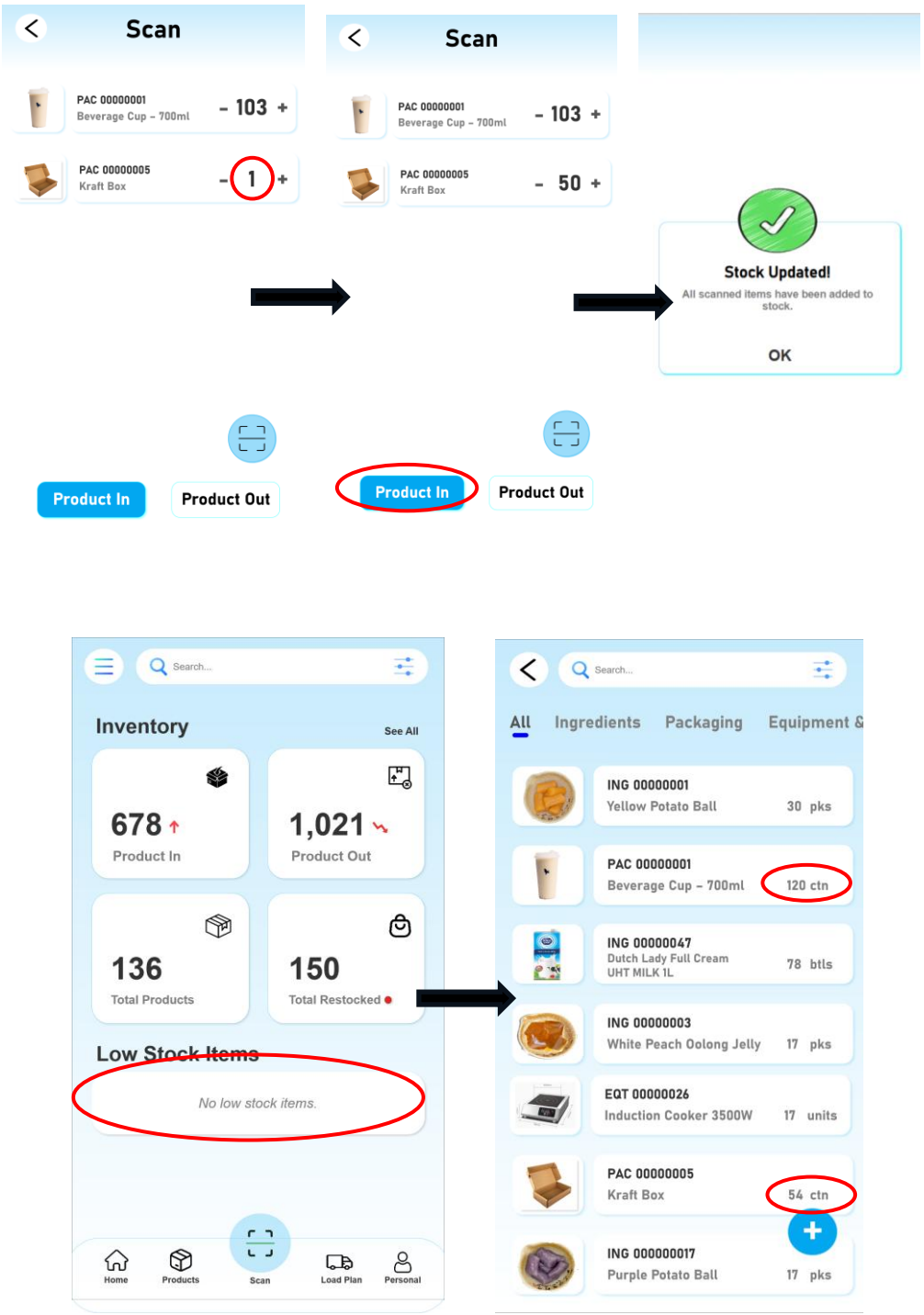



Figure 4. 19: Steps to Update Stock Quantity by Scanning a Barcode.

### 4.6.3 Load Planning Module

The Load Planning Module allows users to generate a load plan based on selected cargo and container types, with the option to add custom containers or cargo sizes if needed. This module also enables users to adjust load plans by dragging and dropping cargo items within the container layout for better optimization. Once the load plan is finalized, users can generate a printable

PDF report that includes a QR code, which can be scanned retrieve the corresponding load plan history. Additionally, the system provides a checklist to verify the arrangement of cargo items, where each item can be marked as “arranged” once it has been correctly loaded.

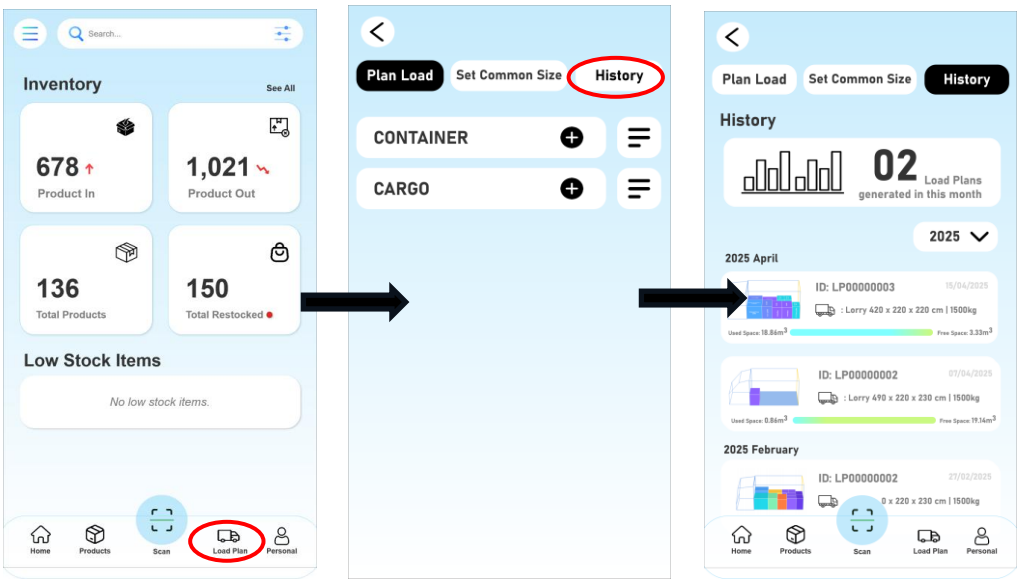



Figure 4. 20: Load Plan History.

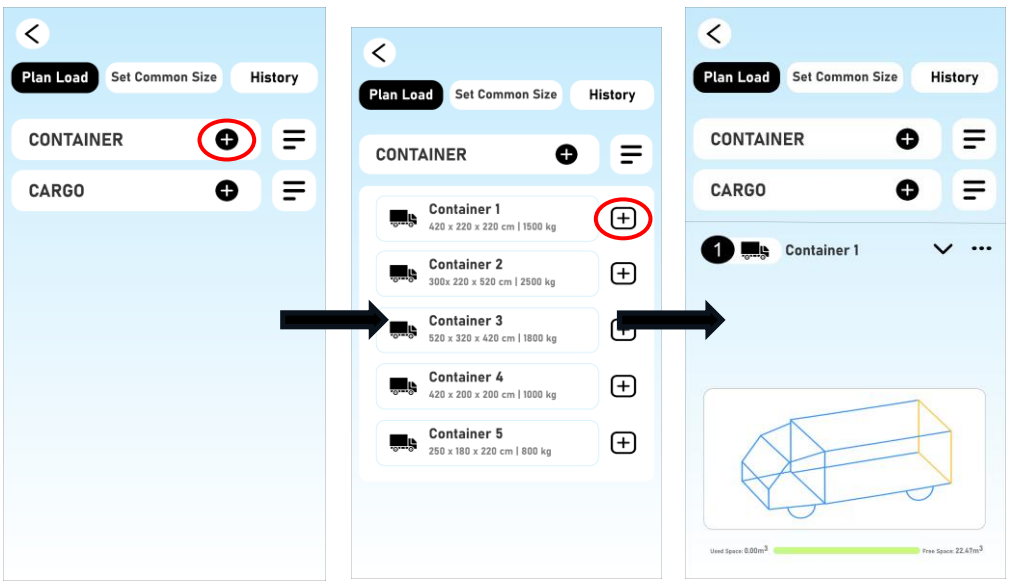



Figure 4. 21: Container Selection Screen.

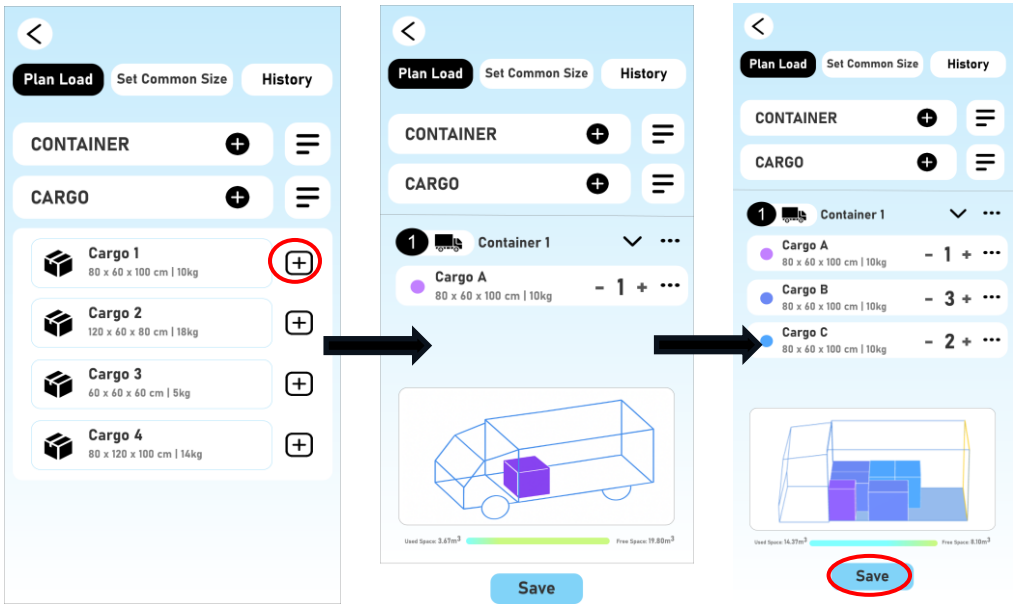



Figure 4. 22: Cargo Selection Screen.

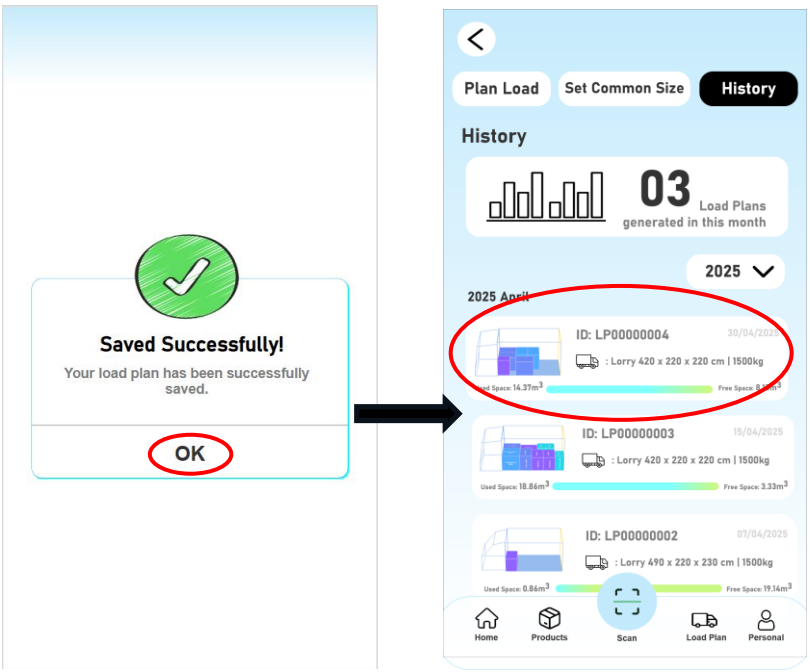



Figure 4. 23: Generated Load Plan and Redirection to Load Plan History.

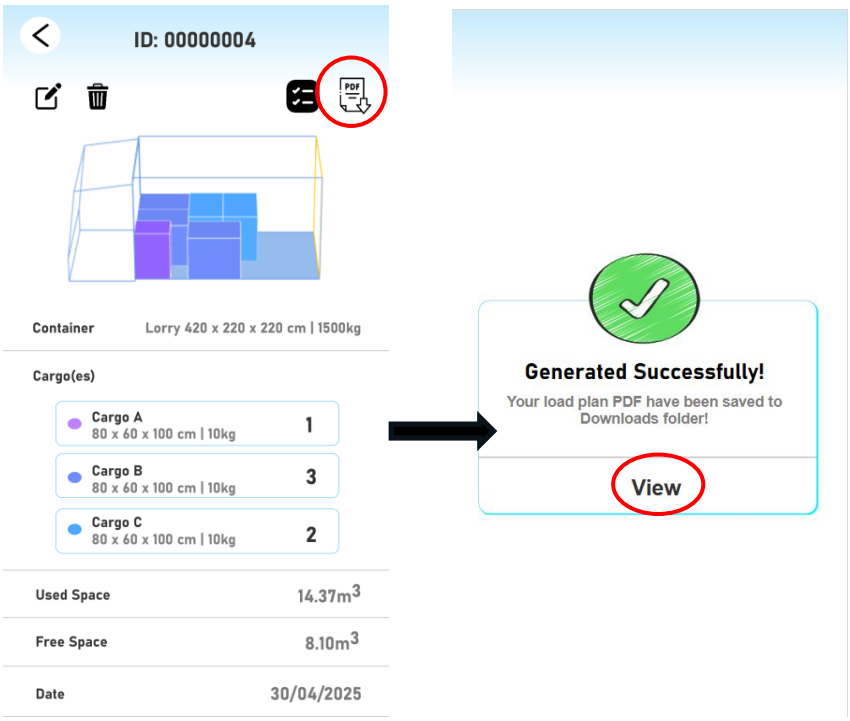



Figure 4. 24: Generating a Printable PDF of the Load Plan.

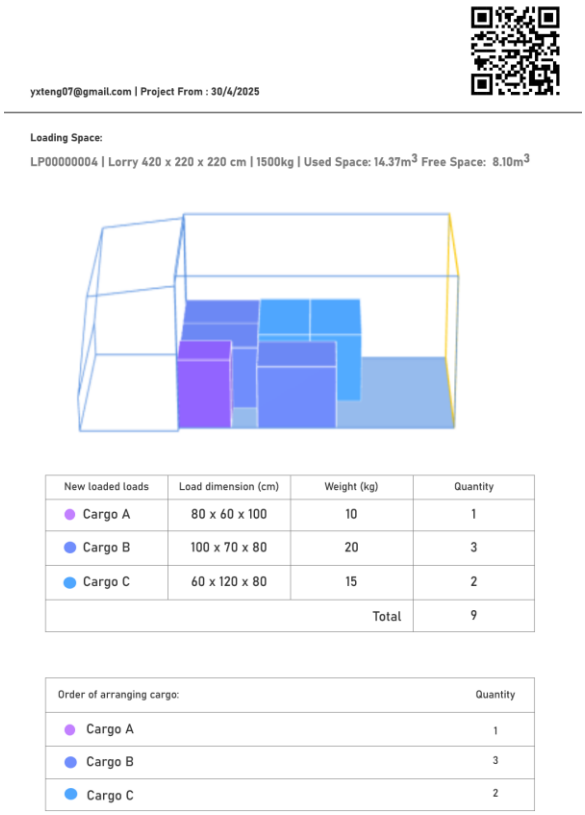



Figure 4. 25: Example of Load Plan PDF Report.

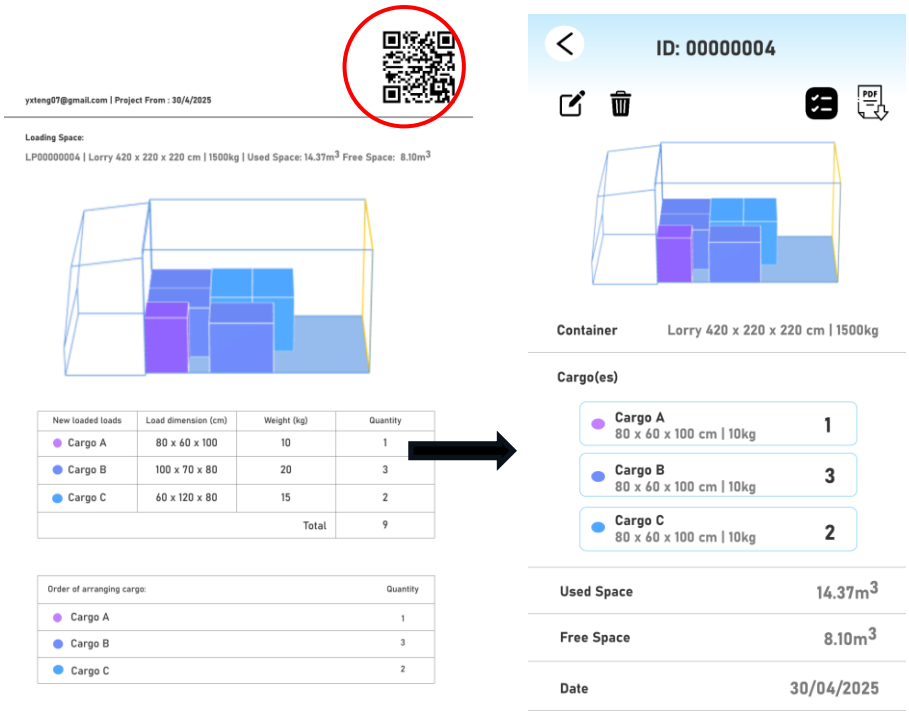



Figure 4. 26: Scanning QR Code on PDF to Retrieve Load Plan Details.

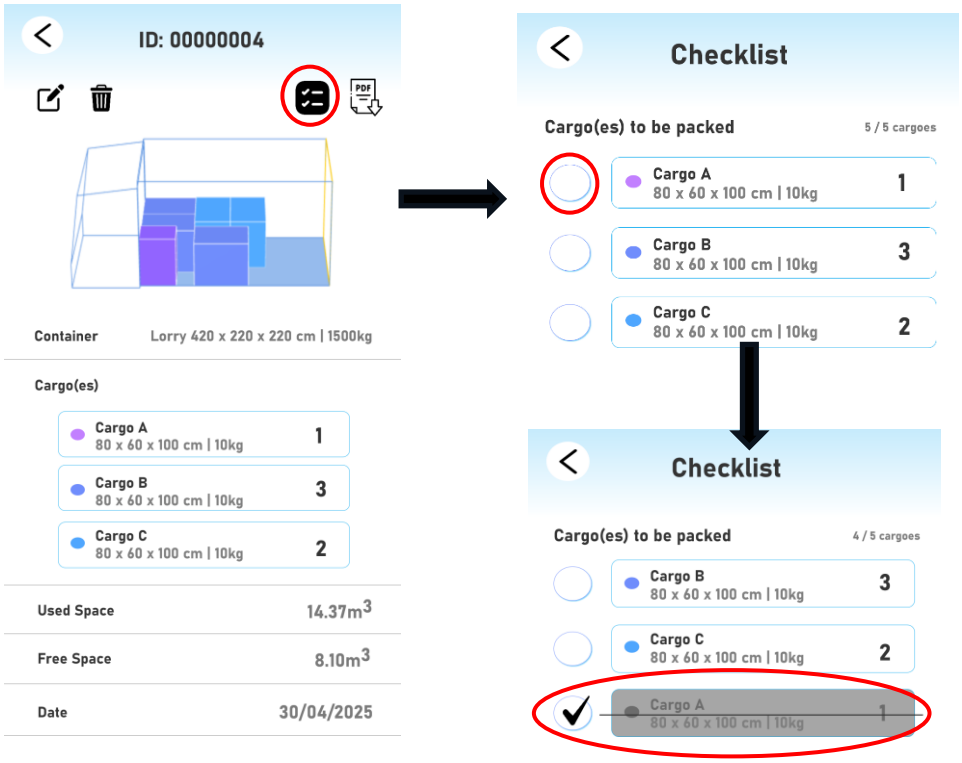



Figure 4. 27: Marking Items as Completed in the Load Plan Checklist.

# CHAPTER 5

# SYSTEM DESIGN

## 5.1 Introduction

This chapter explains the design of the proposed system, outlining how its components are structured and interact to fulfil the required functionalities. The design covers fours main aspects which are system architecture, system database design, activity diagrams and algorithm design.

## 5.2 System Architecture Design

The proposed system architecture follows a three-layer design consisting of the Presentation Layer (Frontend), Application Layer (Backend Services), and Data Layer (Database). The presentation layer provides the graphical user interface that enables users to interact with the system. The application layer acts as the middleware, handling business logic, processing requests, and coordinating communication between the user interface and the database. The data layer is used to store application data and control read and write access to the database, ensuring consistency and security.

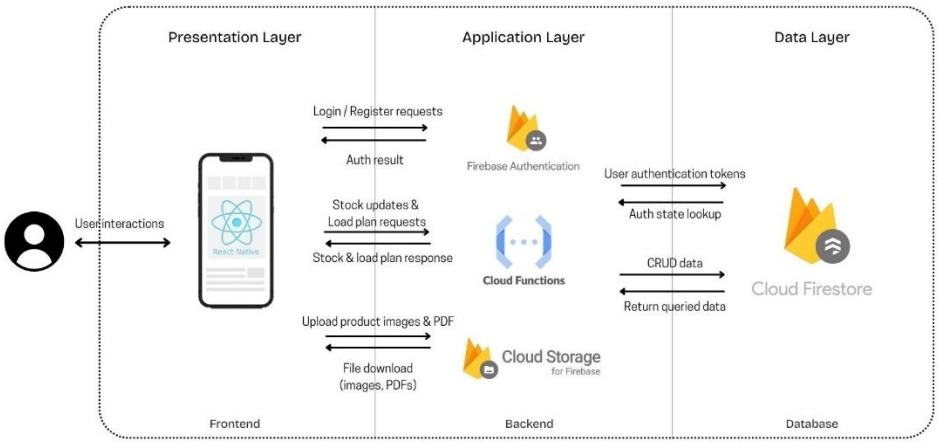



Figure 5. 1: System Architecture Design Diagram

In the proposed mobile application, the Presentation Layer is developed using React Native, which serves as the frontend framework for Android. This layer provides all user-facing interfaces, including login and registration screens, inventory management interfaces, barcode scanning functionality, and load planning visualization. Users interact with the system exclusively through this layer.

The Application Layer is powered by Firebase Backend Services, which handle the core business logic and facilitate communication between the frontend and database. Firebase Authentication manages user login and registration by issuing secure authentication tokens. Cloud Functions act as middleware to process complex logic, such as handling stock update requests, load plan generation, and validating data before committing changes. Cloud Storage is used to upload and retrieve product images, PDFs of load plan reports, and other media files. This layer ensures that all requests from the React Native app are properly validated and routed to the appropriate data services.

The Data Layer consists of Cloud Firestore, a NoSQL database provided by Firebase. Firestore stores structured collections and documents, including inventory records, cargo details, container sizes, and load plan histories. It supports real-time data synchronization, ensuring that updates like stock changes or load plan modifications are instantly reflected across the system. Firestore also enforces access rules, controlling read and write operations to maintain data consistency and security.

## 5.3 System Database Design

### 5.3.1 Entity Relationship Diagram

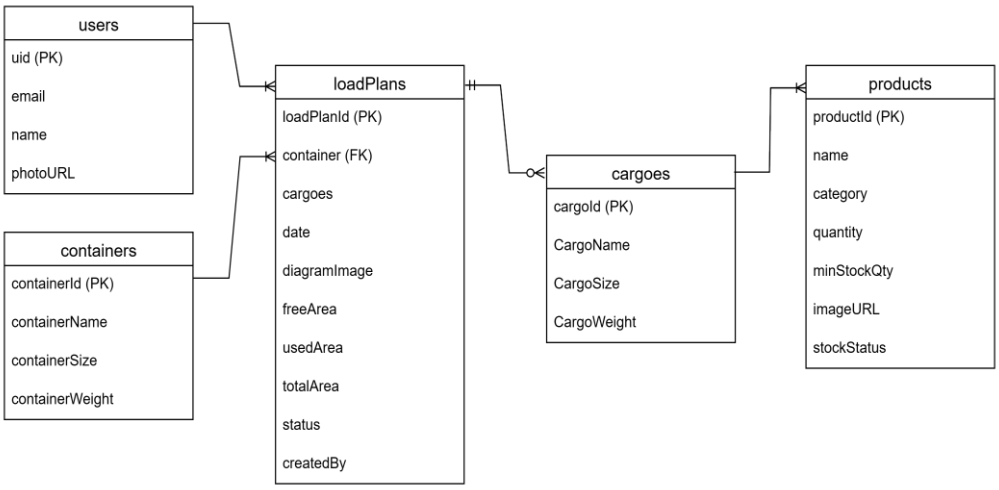



Figure 5. 2: Entity Relation Diagram.

### 5.3.2 Collection Description Table

In the proposed system, data is organized and managed using Firebase Firestore collections. Each collection stores related information in the form of documents, making it easier to organize, retrieve, and maintain.

Table 5. 1: Collection Description Table.

| Collection | Description |
|---|---|
| products | Stores all product details including name, category, current stock quantity, minimum stock quantity, image and stock status. |
| users | Stores all user details, including login information. |
| containers | Stores container details such as dimensions and maximum load capacity. |
| cargoes | Stores cargo item details such dimensions and weight. |
| loadPlans | Stores load plan details, arrangement data, and status. |

### 5.3.3 Data Dictionay

The data dictionary defines the structure and attributes of each collection used in the system. It provides details such as attribute names, descriptions, data types, constraints, and example values.

#### 5.3.3.1 Data Dictionary for products collection

The products collect stores all product details including name, category, current stock quantity, minimum stock quantity, image and stock status.

Table 5. 2: Data Dictionary for products collection.

| Attribute | Description | Data Type | PK/FK | Nullable | Example Values |
|---|---|---|---|---|---|
| productId | Unique identifier for the product generated by Firebase | string | PK | No | PAC979822 |
| name | Product name | string | | No | Beverage Cup 700ml |
| category | Product category | string | | No | Packaging |
| quantity | Current stock quantity | number | | No | 23 |
| minStockQty | Minimum stock quantity before triggering alert | number | | No | 20 |
| imageUrl | URL of product image in Firebase Storage | string | | No | https://firebasestorage.googleapis.com/v0/b/inventoryapp5c48a.firebasestorage.app/o/ |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | productImages%2F1756528824262.jpg?alt=media&token=49ea3d32-0118-4654-8956-6ff803b6c14b |
| stockStatus | Current stock status | string | | No | In Stock |

#### 5.3.3.2 Data Dictionary for users collection

User collection stores all user details, including login information.

Table 5. 3: Data Dictionary for users collection.

| Attribute | Description | Data Type | PK/FK | Nullable | Example Values |
|---|---|---|---|---|---|
| uid | Unique identifier of user generated by Firebase Authentication | string (Auth ID) | PK | No | nV719KbgkaQc5sLvFOz9mQW04Rp2 |
| email | User's registered email address | string | | No | ali@gmail.com |
| name | User's display name | string | | No | Ali |
| photoURL | URL of user's profile photo (empty if not uploaded) | string | | Yes | " " |

#### 5.3.3.3 Data Dictionary for containers collection

The containers collection stores container details such as dimensions and maximum load capacity.

Table 5. 4: Data Dictionary for containers collection.

| Attribute | Description | Data Type | PK/FK | Nullable | Example Values |
|---|---|---|---|---|---|
| containerId | Unique identifier of container generated by Firestore | string | PK | No | 2DyYRuG8mnwTZnM7bMKC |
| ContainerName | Name of the container | string | | No | Container 1 |
| ContainerSize | Dimensions of the container (L×W×H in cm) | string | | No | 420x200x220cm |
| ContainerWeight | Weight capacity of the container | string | | No | 1500kg |

#### 5.3.3.4 Data Dictionary for cargoes collection

Cargoes collection stores cargo item details such dimensions and weight.

Table 5. 5: Data Dictionary for cargoes collection.

| Attribute | Description | Data Type | PK/FK | Nullable | Example Values |
|---|---|---|---|---|---|
| cargoId | Unique identifier of cargo generated by Firestore | string | PK | No | VKXOd8ZUnTf3Tgmj4EAP |
| CargoName | Name of the cargo item | string | | No | Cargo 1 |
| CargoSize | Dimensions of cargo (L×W×H in cm) | string | | No | 80×60 ×100 cm |
| CargoWeight | Weight of the cargo | string | | No | 10kg |

#### 5.3.3.5 Data Dictionary for loadPlans collection

The loadPlans collection stores load plan details, arrangement data, and status.

Table 5. 6: Data Dictionary for loadPlans collection.

| Attribute | Description | Data Type | PK/FK | Nullable | Example Values |
|---|---|---|---|---|---|
| loadPlanId | Unique identifier for the load plan | string | PK | No | 2 |
| cargoes | List of cargo items included in this | array of objects | | No | [ {CargoName: "Cargo 2", CargoSize: |

| | | | | | |
|---|---|---|---|---|---|
| | load plan | | | | "100x100x140cm", CargoWeight: "10kg", quantity: 3}, … ] |
| container | Container details used in the plan | object | FK | No | {ContainerName: "Container 1", ContainerSize: "420x200x220cm", ContainerWeight: "1500kg"} |
| date | Date and time when the load plan was created | timestamp | | No | 2025-09-12T06:38:48.899Z |
| diagramImage | Cargo arrangement diagram image | string | | No | iVBORw0KGgoAAAANSUhEUgAAA… |
| freeArea | Remaining free area in container | number | | No | 2.05 |
| usedArea | Used area in container | number | | No | 6.35 |
| totalArea | Total area of container | number | | No | 8.4 |
| status | Status of the load plan | string | | No | finished/pending |

## 5.4 Activity Diagram

The Activity Diagrams provide a visual representation of the workflows in the proposed system. They illustrate the sequence of user actions and system responses for various use cases, ensuring a clear understanding of process flows, decision points, and interactions between the user and the system.

### 5.4.1 Register account Activity Diagram

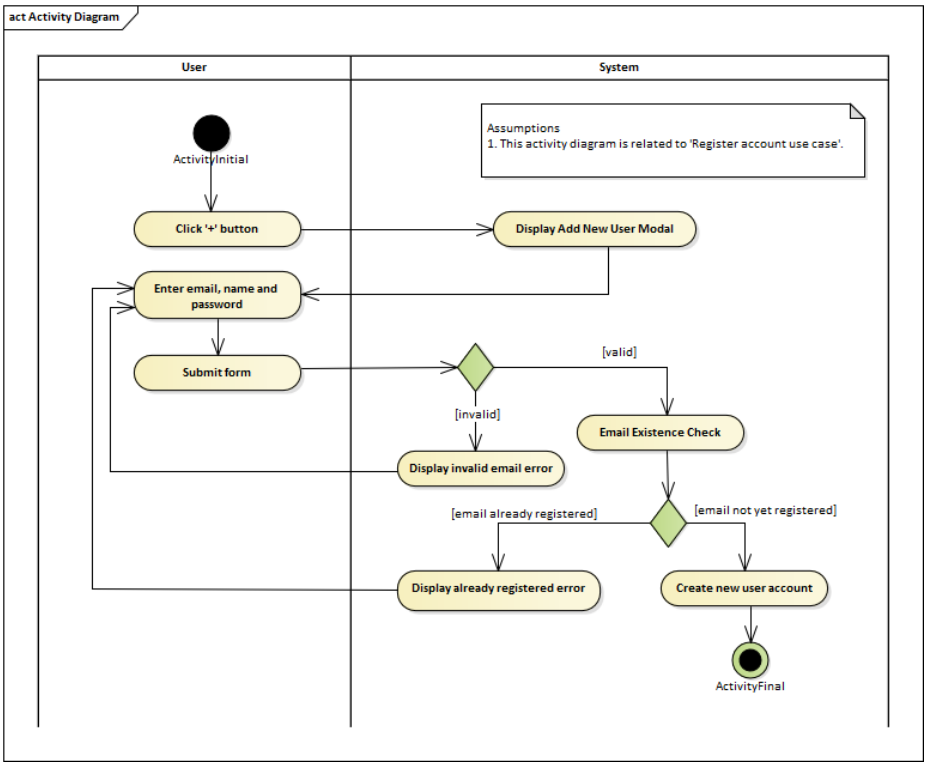



Figure 5. 3: Register account Activity Diagram.

### 5.4.2 Login account Activity Diagram

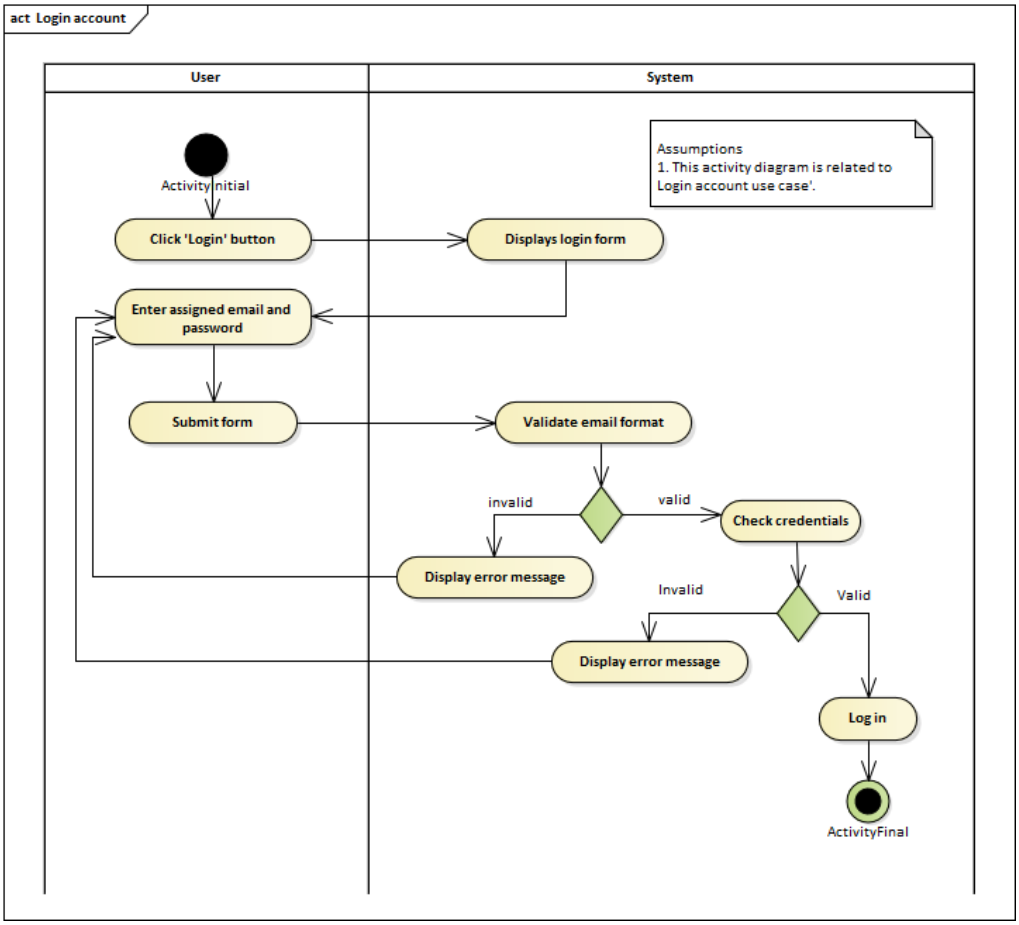



Figure 5. 4: Login account Activity Diagram.

### 5.4.3 Scan item barcode Activity Diagram

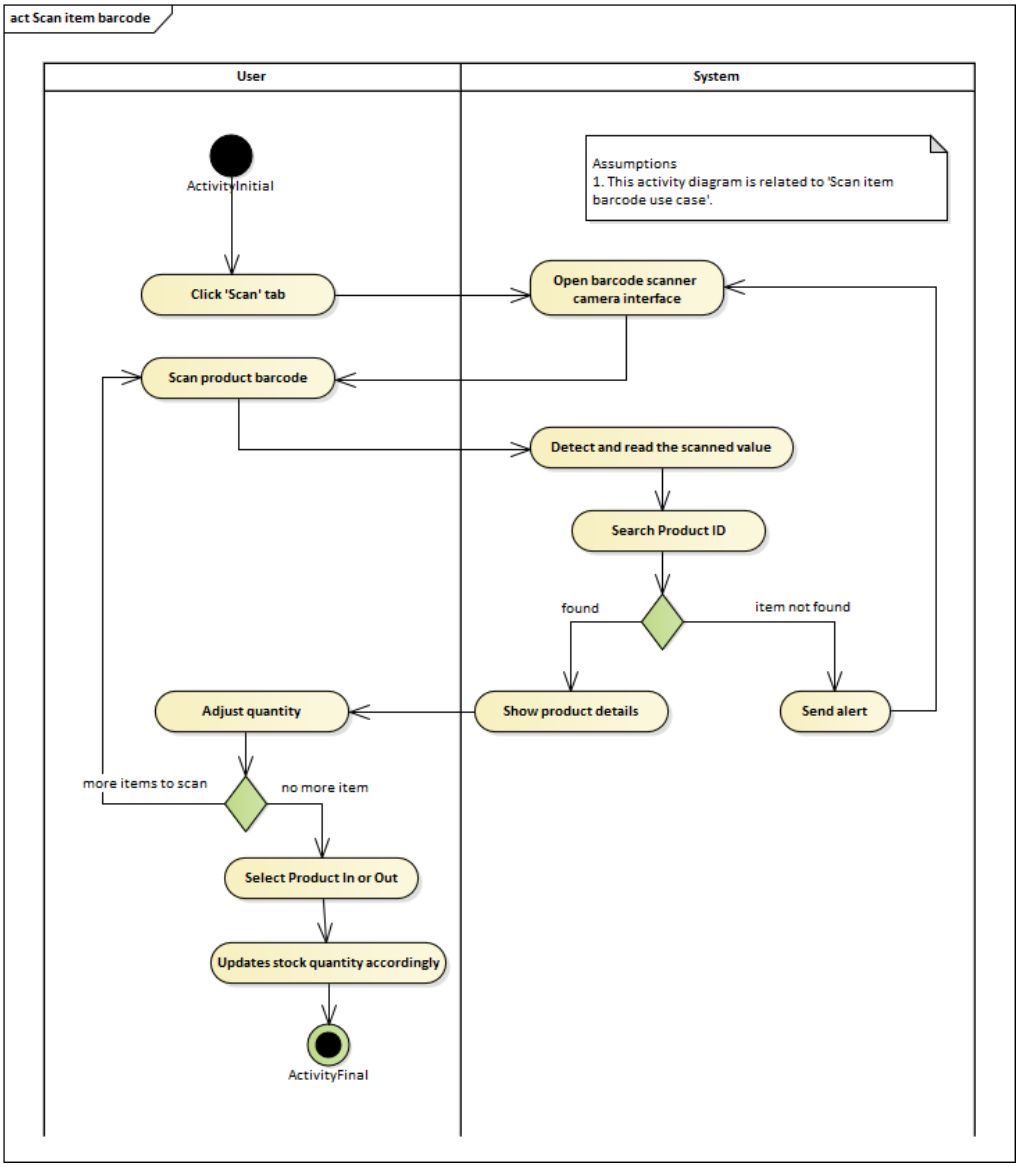



Figure 5. 5: Scan item barcode Activity Diagram.

### 5.4.4 Update stock quantity activity diagram

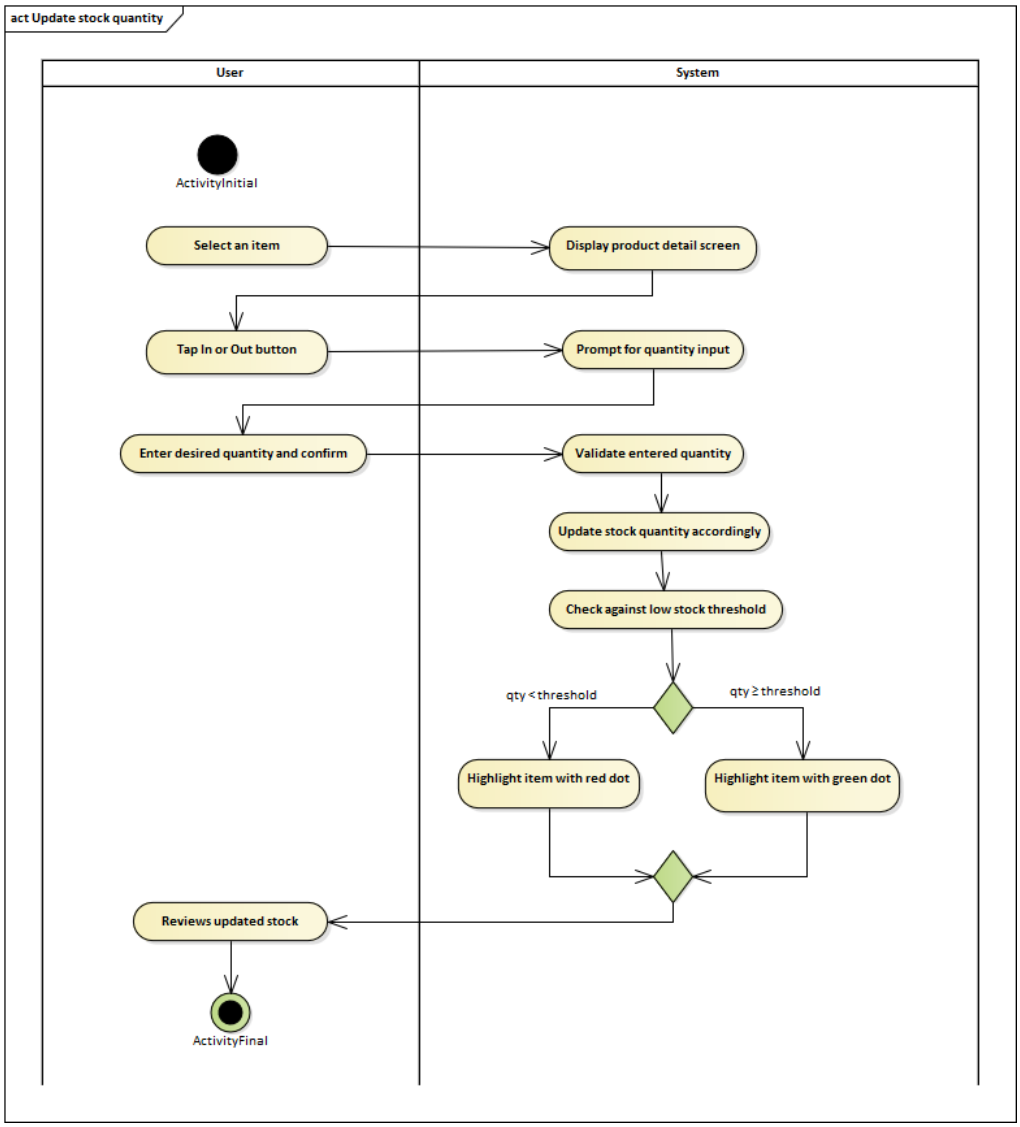



Figure 5. 6: Update stock quantity activity diagram.

### 5.4.5 View inventory list activity diagram

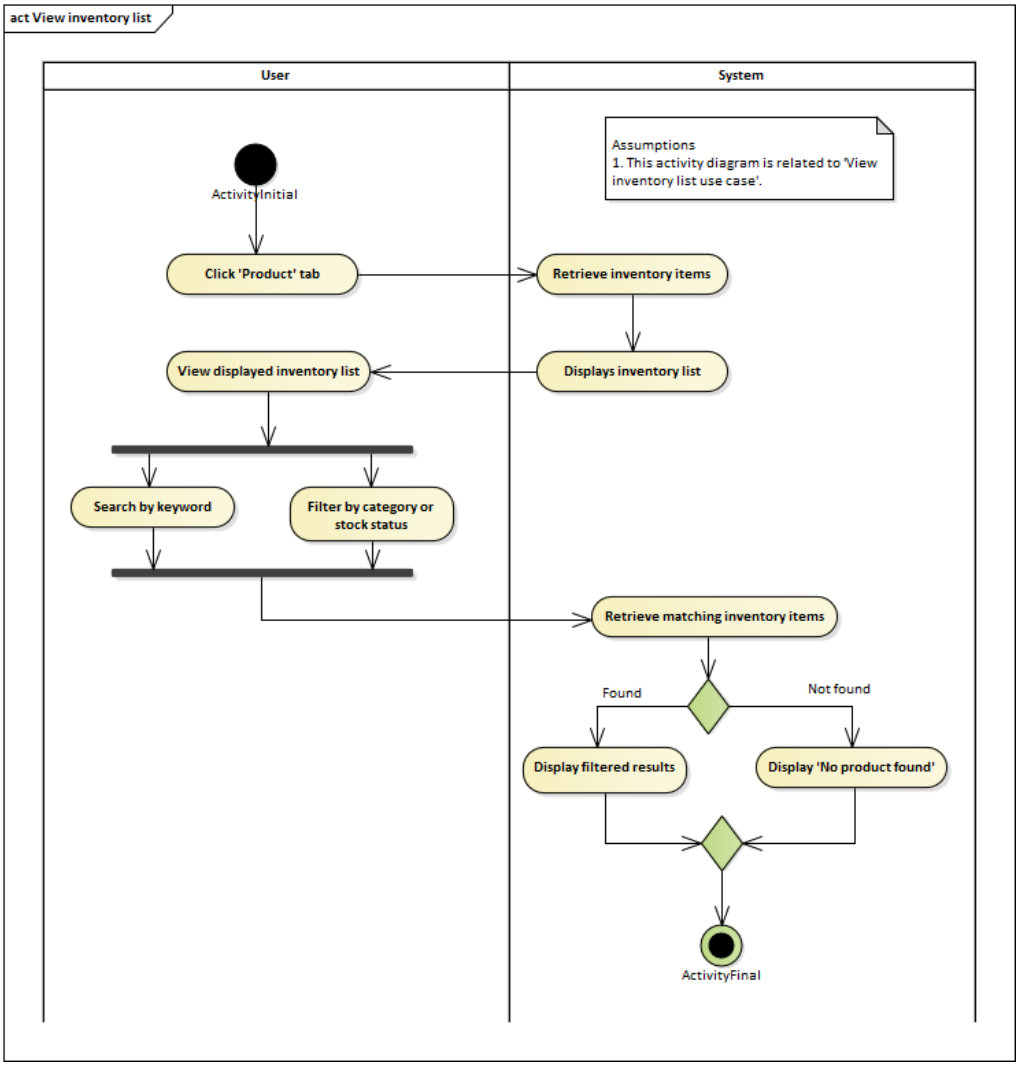



Figure 5. 7: View inventory list activity diagram.

### 5.4.6 Add new item activity diagram

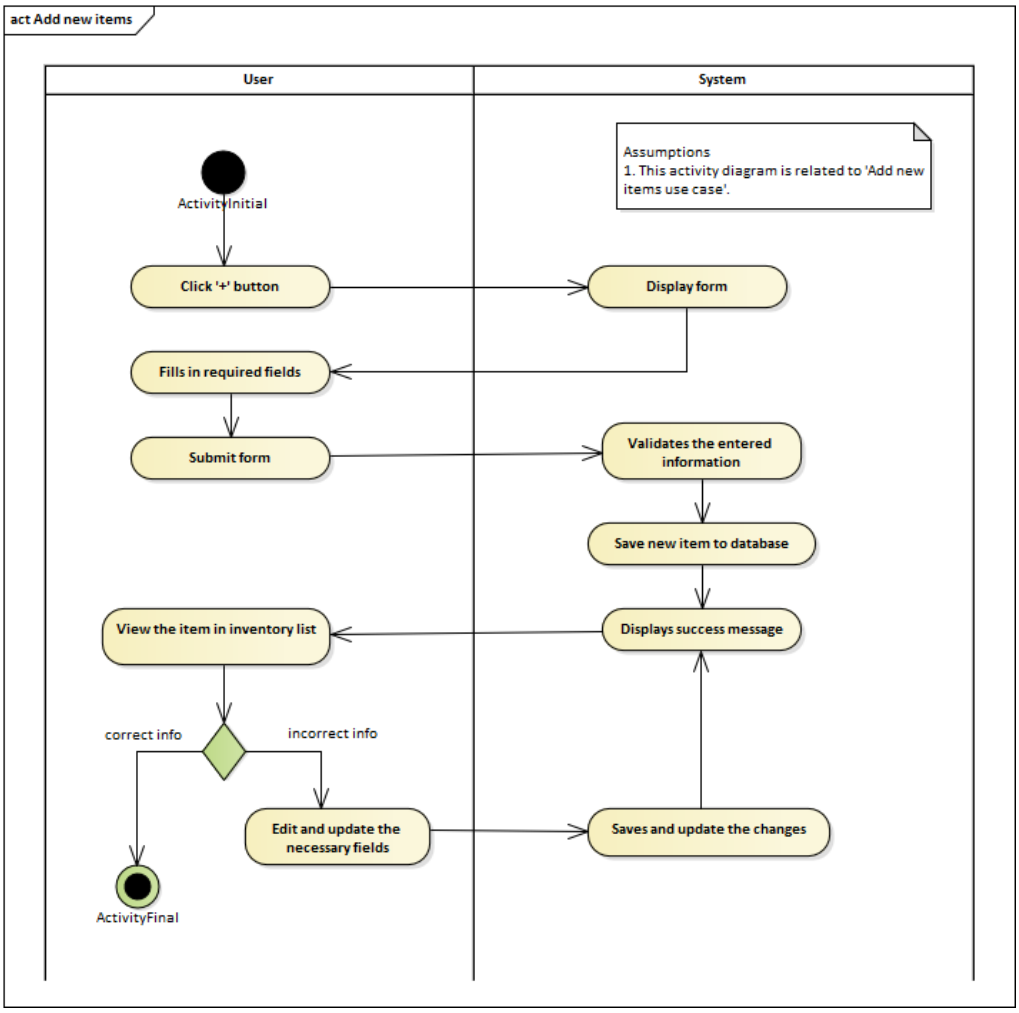



Figure 5. 8: Add new item activity diagram.

### 5.4.7 Delete inventory items Activity diagram

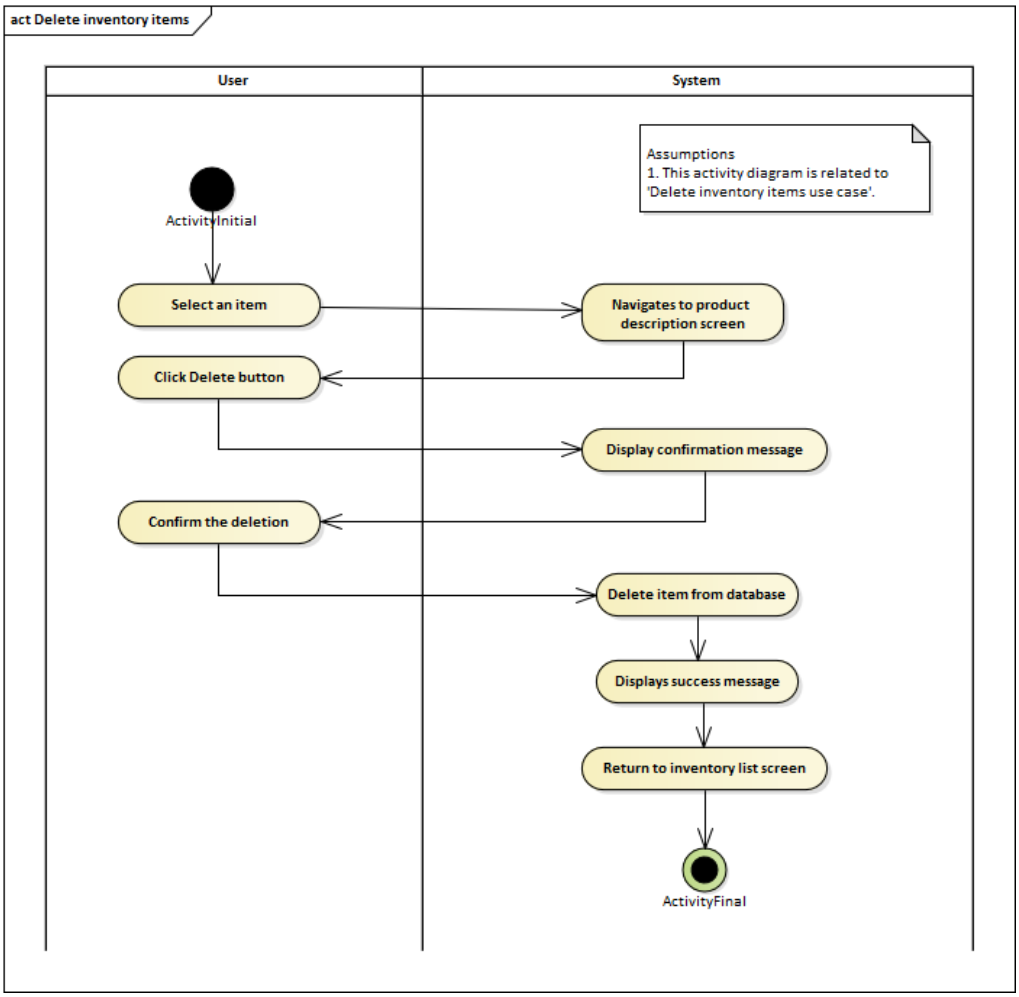



Figure 5. 9: Delete inventory items Activity diagram.

### 5.4.8 Generate Load Plan Activity diagram

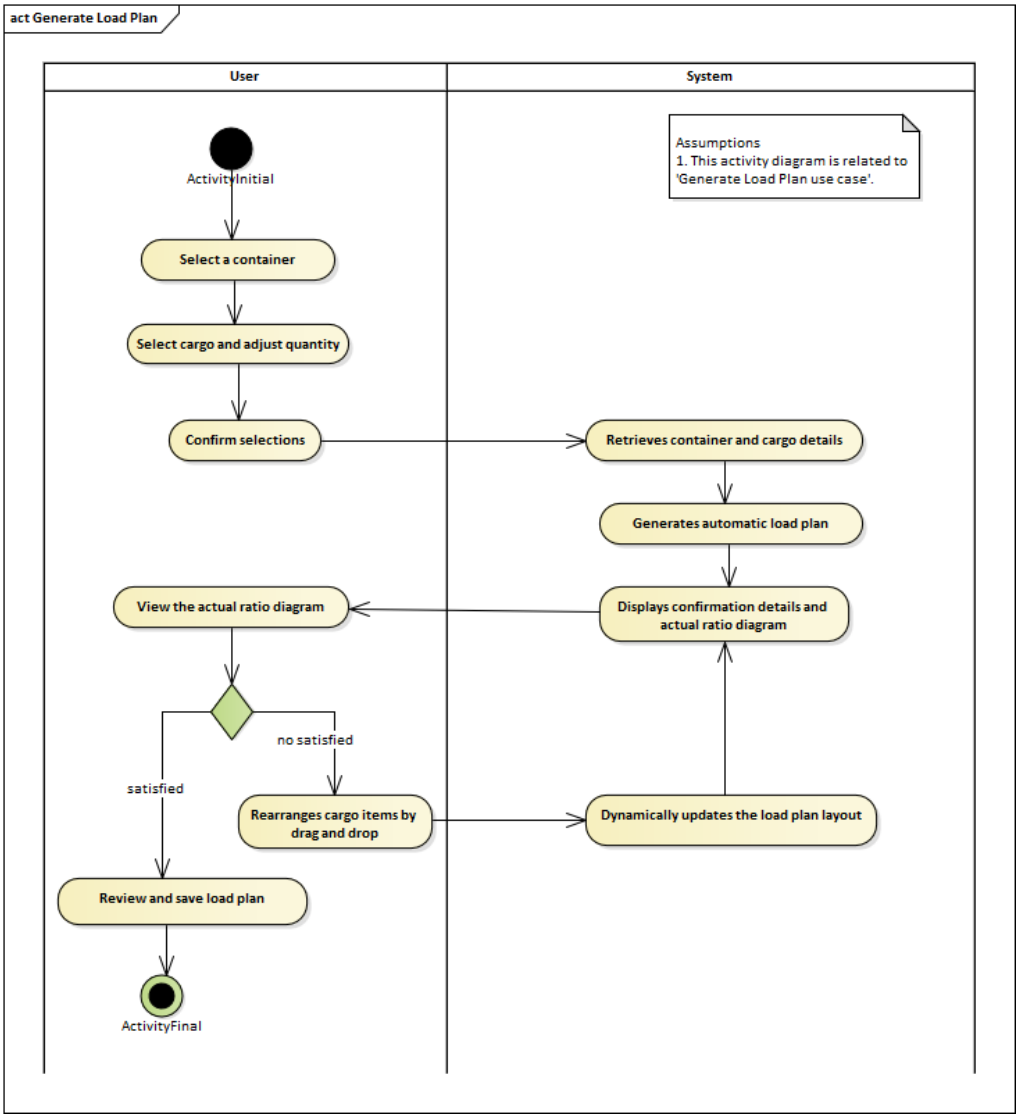



Figure 5. 10: Generate Load Plan Activity diagram.

### 5.4.9 View the checklist Activity diagram

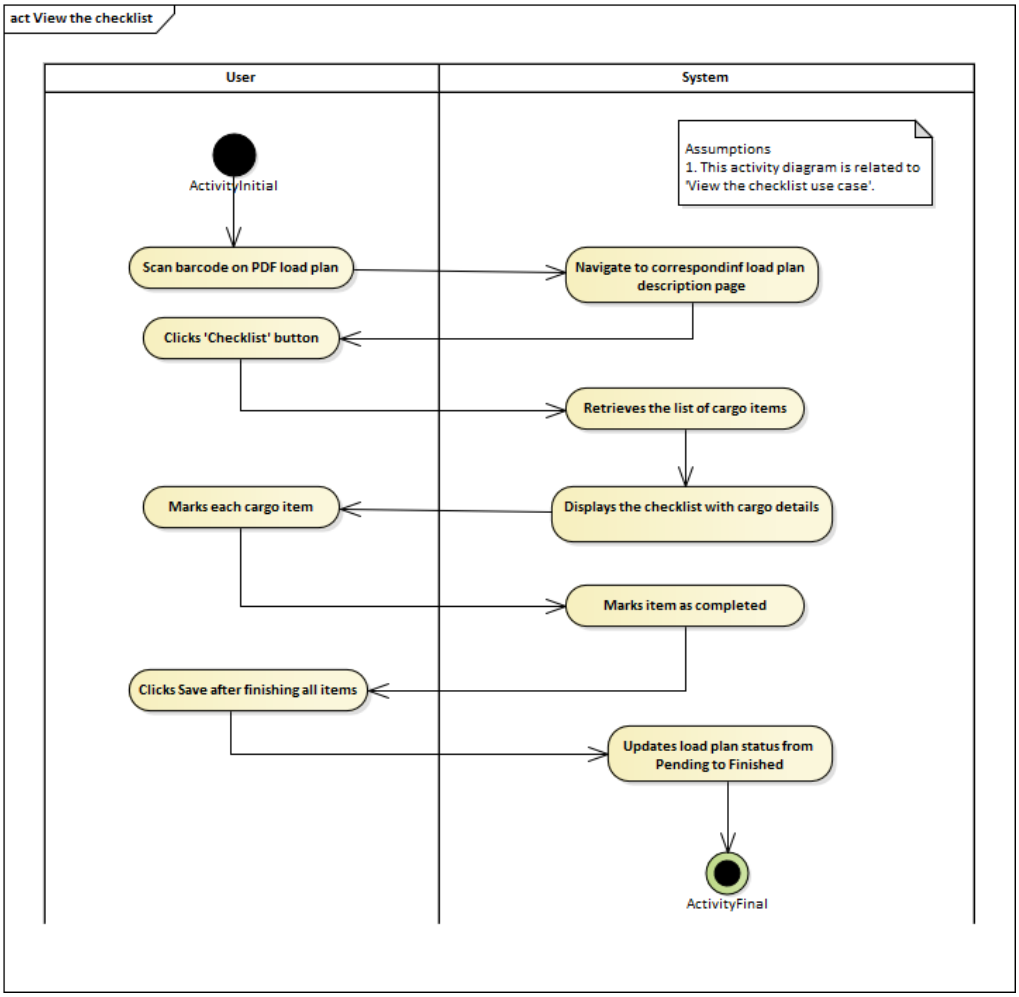



Figure 5. 11: View the checklist Activity diagram.

### 5.4.10 Generate PDF report Activity diagram

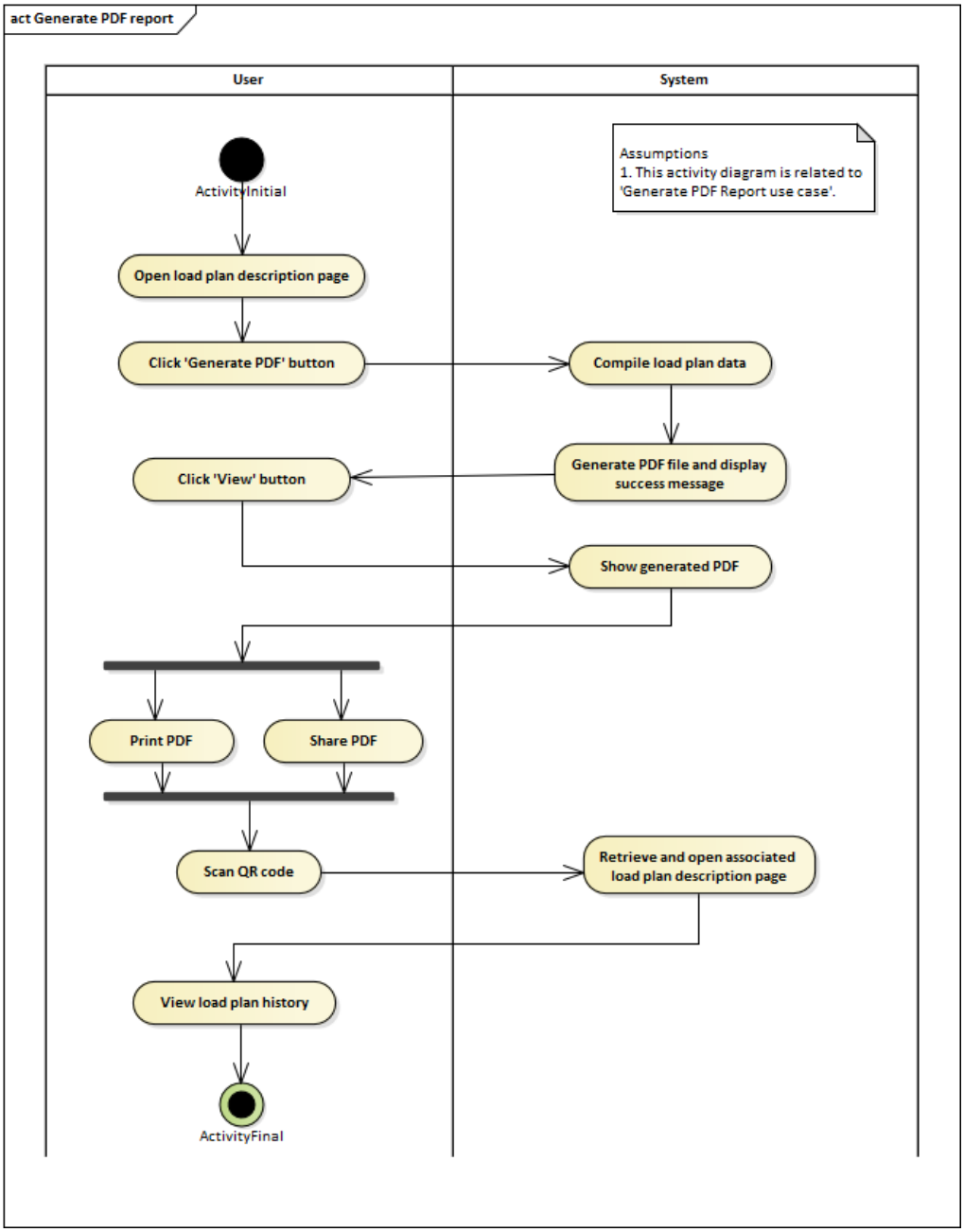



Figure 5. 12: Generate PDF report Activity diagram.

## 5.5 Algorithm Design

This section presents the algorithm applied in the system to optimize the cargo arrangement process. The Binary Tree Bin Packing algorithm was chosen as it provides an efficient way to place items within a container by recursively partitioning available space. The following subsections describe the concept, flow, pseudocode, and flowchart of the algorithm used in this project.

### 5.5.1 Algorithm Concept

The algorithm implemented in this project is the Binary Tree Bin Packing algorithm, which is designed to efficiently arrange rectangular cargo items within a container while maximizing space utilization. The concept is based on representing the container as a binary tree, where each node corresponds to a rectangular space available for item placement.

When a cargo item is placed into a node, that space is marked as used and split into two child nodes. One representing the space to the right of the item and the other representing the space below the item. This type of subdivision is known as the Guillotine split, where space is recursively divided into two smaller rectangular regions along straight horizontal or vertical lines. Such recursive partitioning ensures that all available areas within the container are systematically tracked and reused for subsequent items.

The algorithm processes cargo items sequentially, typically sorted in descending order by their area to prioritize placement of larger items first. For each item, the algorithm searches for a suitable node that can accommodate its dimensions. If a fitting node is found, the item is placed, and the space is subdivided. If no node can accommodate the item, it is classified as an unfit item.

Unlike some general implementations of Binary Tree Bin Packing that may allow item rotation, the algorithm used in this project is restricted to a 2D, non-rotational Guillotine split approach. This constraint was chosen to maintain simplicity in the algorithm's design and execution, ensuring that the solution remains lightweight and efficient. Such a simplification is sufficient for small and medium-sized enterprises (SMEs), where the focus is on ease of use and practicality rather than handling highly complex packing scenarios. By

avoiding rotation, the algorithm preserves the natural orientation of cargo items, making the loading process more straightforward for staff while still maximizing space utilization.

By recursively splitting and tracking available regions, the Binary Tree Bin Packing algorithm provides a balance between computational efficiency and effective use of container space, making it well-suited for applications in inventory management and cargo load planning.

### 5.5.2 Algorithm Flow

Table 5. 7: Algorithm Flow.

| No | Step | Description |
|---|---|---|
| 1 | Input Preparation | The algorithm receives the container dimensions and cargo list. Cargo dimensions are parse to {length, width, height} from strings such as "420x220x200". |
| 2 | Expand Cargo List | Each cargo with a given quantity is expanded into individual cargo items (flatCargoes), each with a unique index (_flatIdx). |
| 3 | Convert to 2D Rectangles | Each cargo item is represented as a rectangle (w, h), where w = length and h = width. Two versions are generated which are in centimeters (for fit/unfit checking) and in pixels (for UI rendering). |
| 4 | Sort by Area | The cargo items are sorted in descending order by area (w × h) so that larger items are placed first, increasing packing efficiency. |
| 5 | Create Root Node | The container is represented as the root node of a binary tree, (x=0, y=0, w=containerLength, h=containerWidth). Initially, it is unused and has no children. |
| 6 | Iterative Placement | For each cargo item, the algorithm attempts to find a suitable node. |
| 6.1 | Find Node | Call btFindNode(node, w, h):<br>· If node.used = true, recursively search |

| | | |
|---|---|---|
| | | node.right and node.down.<br>· If the item fits within the node (w ≤ node.w and h ≤ node.h), return the node.<br>· Otherwise, return null. |
| 6.2 | Placement Decision | If a suitable node is found, place the item.<br>If no node fits, add the item to the unfit list. |
| 6.3 | Place Item | Place the cargo at (node.x, node.y). Mark the node as used. |
| 6.4 | Split Space | Apply a guillotine split to divide remaining free space.<br>· Right node, space to the right of the placed item.<br>· Down node, space below the placed item. |
| 6.5 | Record Result | Store the placement in the placements map (for fit items) or in the unfit list (for items that cannot fit). |
| 7 | Continue Loop | Repeat steps 6.1–6.5 until all cargo items are processed. |
| 8 | Output Generation | The algorithm returns two results which are<br>· Placements, list of successfully placed cargo items with coordinates.<br>· Unfit Items, list of items that could not fit into the container. |
| 9 | Visualization | In the UI, successfully placed items are drawn on the diagram at their coordinates. Unfit items are listed below as “Not Fit Into Container.” |

### 5.5.3 Pseudocode

The pseudocode describes how the Binary Tree Bin Packing algorithm arranges cargo items within the container. Each item is processed individually, starting with the largest by area. The algorithm searches for a suitable free node within the container space using a recursive function that explores the right and down child nodes until a fit is found, or no space is available. When an item fits, it is placed at the node's coordinates, and the node is split into two sub-nodes to represent the remaining free space. If no suitable node is found, the item is classified as unfit. This recursive placement and splitting process continues until all items are considered, resulting in a final load plan layout with clearly identified placed and unfit items.

Figure 5. 13: Pseudocode of Binary Bin Packing algorithm.

```
Input:
   cargoList – list of cargo items (length, width, quantity)
   container – container dimensions (length, width)

Output:
   placements – coordinates of placed items
   unfitItems – list of items that cannot fit

Begin
   1. Expand cargoList so that each unit of quantity is represented as an
      individual item.
   2. Convert each cargo item into a 2D rectangle (w, h).
   3. Sort cargo items in descending order of area (w × h).
   4. Create the root node representing the entire container space.
   5. For each cargo item in cargoList do
      5.1 Attempt to find a node for the item:
         If node.used = true then
            recursively check node.right or node.down
            nodeFound ← FindNode(node.right, item.w, item.h) ||
                        FindNode(node.down, item.w, item.h)
```

```
        Else if (item.w ≤ node.w) and (item.h ≤ node.h) then
            nodeFound ← node
        Else
            nodeFound ← null

    5.2 If nodeFound ≠ null then
            Place item at (node.x, node.y)
            node.used ← true
            Split remaining space into:
                1. Down node → (x = node.x, y = node.y + item.h,
                                w = node.w, h = node.h − item.h)
                2. Right node → (x = node.x + item.w, y = node.y,
                                w = node.w − item.w, h = item.h)
            Add placement to placements
        Else
            Add item to unfitItems
        End If
    End For
    6. Return placements and unfitItems
End
```

### 5.5.4 Flowchart

The flowchart illustrates the Binary Tree Bin Packing algorithm applied for arranging cargo items in a container. The process begins by taking the container dimensions and a list of cargo items defined by their length, width, and quantity. The cargo list is expanded by quantity, converted into 2D rectangles, and sorted by area in descending order. A root node is then created to represent the available container space. For each cargo item, the algorithm recursively searches for a suitable node by checking if the node is unused and large enough to accommodate the item. If a valid node is found, the item is placed at the node's coordinates, the node is marked as used, and the remaining space is split into right and down sub-nodes. The placement is then recorded. If no valid node is found, the item is added to the unfit list. This process continues until all items are processed, after which the algorithm outputs both the successful placements and the list of unfit items.

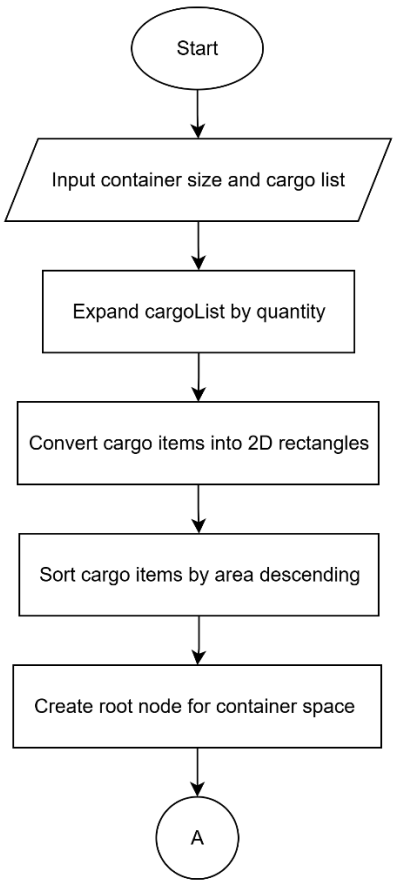



Figure 5. 14: Flowchart of Binary Tree Bin Packing Algorithm (Part 1).

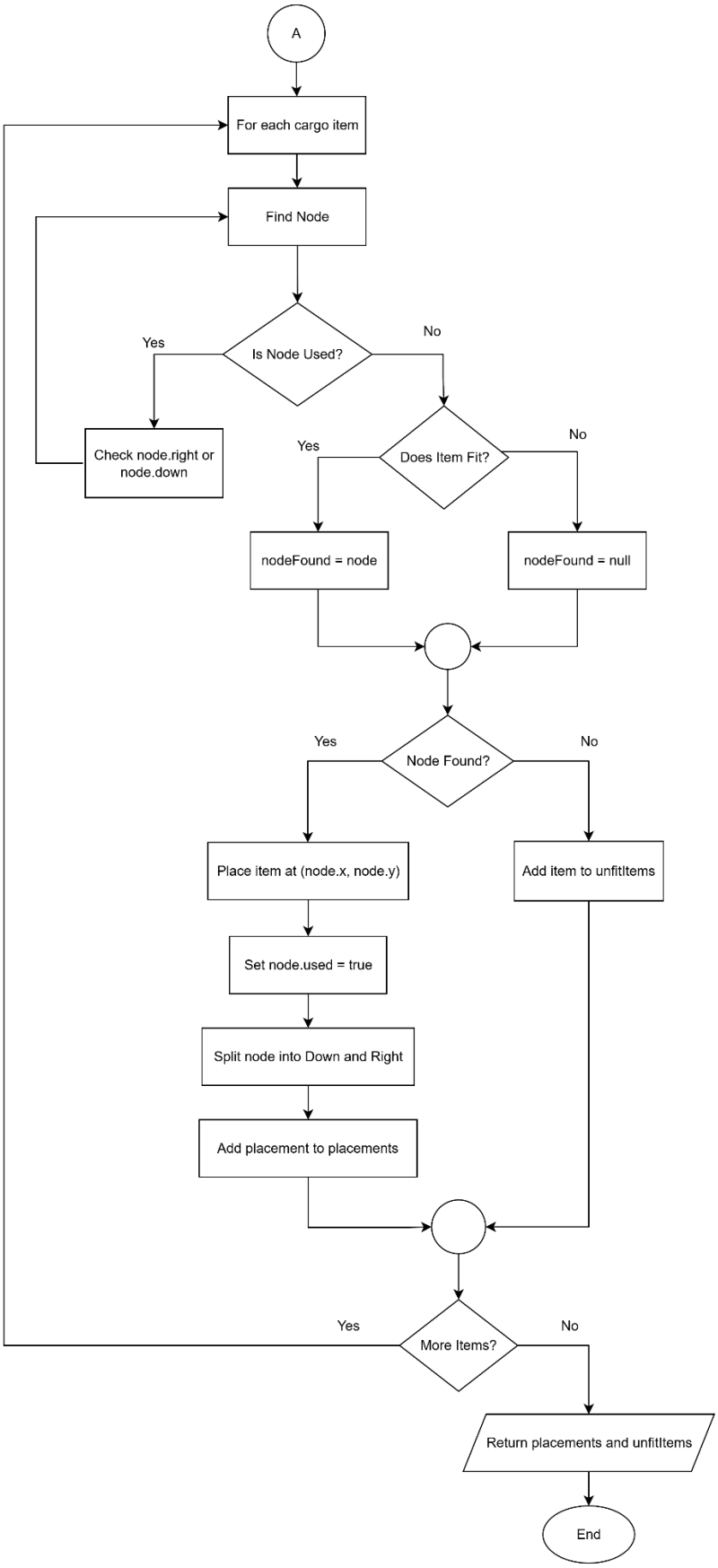



Figure 5. 15: Flowchart of Binary Tree Bin Packing Algorithm (Part 2).

### 5.5.5 Traceability Table of Flowchart, Pseudocode and Implementation Code

Table 5. 8: Traceability Table of Flowchart, Pseudocode and Implementation Code.

| **Flowchart Step** | **Pseudocode Statement** | **Implementation** |
|---|---|---|
| Input container size and cargo list | Input: cargoList, container | Function getFitStatus2D(cargoes, container) and getInitialPositions(cargoes, container) parse inputs |
| Expand cargoList by quantity | Expand cargoList | flatCargoes = cargoes.flatMap(cargo => Array(cargo.quantity).fill({...})) |
| Convert to 2D rectangles | Convert each cargo into (w,h) | const itemsPx = flatCargoes.map((cargo, idx) => {<br>const s = parseSize(cargo.CargoSize);<br>return { _flatIdx: idx, w: s.length * scaleLength, h: s.width * scaleWidth, orig: idx };<br>}); |
| Sort by area descending | Sort cargo items by w × h | const sorted = [...itemsPx].sort((a, b) => (b.w * b.h) - (a.w * a.h)); |
| Create root node | root ← {x=0,y=0,w,h,used=false} | const root = { x: 0, y: 0, w: binW, h: binH, used: false, right: null, down: null } in btPackFixedBin |
| Find Node | nodeFound ← FindNode(node.right,w,h) | |

| | | |
|---|---|---|
| Does item fit? | if (w ⩽ node.w and h ⩽ node.h) | const btFindNode = (node, w, h) => {<br>if (!node) return null;<br>if (node.used) {<br>return btFindNode(node.right, w, h) \|\| btFindNode(node.down, w, h);<br>}<br>if (w <= node.w && h <= node.h) return node;<br>return null;<br>}; |
| Place item | Place item at (node.x, node.y) | placements.set(it._flatIdx, { x: usedNode.x, y: usedNode.y, w: it.w, h: it.h }); |
| Mark node as used | node.used ← true | const btSplitNode = (node, w, h) => {<br>node.used = true; |
| Split node into Right and Down | split into right/down nodes | const btSplitNode = (node, w, h) => {<br>node.used = true;<br>node.down = { x: node.x, y: node.y + h, w: node.w, h: node.h - h, used: false, right: null, down: null };<br>node.right = { x: node.x + w, y: node.y, w: node.w - w, h: h, used: false, right: null, down: null }; |

| | | |
|---|---|---|
| | | return node;<br>}; |
| Add placement | Add placement to placements | const btPackFixedBin = (items, binW, binH) => {<br>const root = { x: 0, y: 0, w: binW, h: binH, used: false, right: null, down: null };<br>const placements = new Map();<br>const unplacedIdx = [];<br><br>for (const it of items) {<br>const node = btFindNode(root, it.w, it.h);<br>if (node) {<br>const usedNode = btSplitNode(node, it.w, it.h);<br>placements.set(it._flatIdx, { x: usedNode.x, y: usedNode.y, w: it.w, h: it.h });<br>} else {<br>unplacedIdx.push(it._flatIdx);<br>}<br>} |
| Unfit item | Add item to unfitItems | |

| | | |
|---|---|---|
| | | return { placements, unplacedIdx };<br>}; |
| Loop (More items?) | for each cargo item | const itemsSorted = [...items].sort((a, b) => (b.w * b.h) - (a.w * a.h)); |
| Return placements & unfitItems | return placements, unfitItems | return { fitCargoes, unfitCargoes: mergedUnfit }; from getFitStatus2D<br>return positions; from getInitialPositions |

## 5.6 Conclusion

In conclusion, this chapter has presented the overall system design of the proposed mobile application. The design was structured into several sections, including system architecture design, database design, activity diagrams, and algorithm design. The architecture design described the three-layer structure of the application, consisting of the presentation, application, and data layers. The database design detailed the organization of data using Firebase Firestore collections, supported by collection description tables and data dictionaries. Furthermore, activity diagrams were used to illustrate the flow of key system functionalities, offering clear visualizations of user–system interactions. Additionally, the algorithm design section described the implementation of the Binary Tree Bin Packing algorithm (2D, Guillotine split, no rotation), which serves as the core mechanism for optimizing cargo arrangement within the container.

# CHAPTER 6

# SYSTEM IMPLEMENTATION

## 6.1 Introduction

This chapter focuses on transforming the design specifications into a functional mobile application. In this phase, all modules and components that were designed in the previous chapter are developed and integrated to ensure the system performs as intended.

## 6.2 Development Environment Setup

Both hardware and software environments were properly configured before implementation.

### 6.2.1 Hardware Requirements

The mobile application was developed using a personal computer equipped with an Intel i9 processor, 16 GB of RAM, and the Windows 11 operating system. To validate the application on a real device and ensure compatibility with real-world usage, an Android smartphone running Android 10 or higher was used for testing. Alternatively, in the absence of a physical device, the Android Studio emulator with the Pixel 4a API 30 configuration served as an option for debugging and functional testing throughout the development process.

### 6.2.2 Software Requirements

The system implementation required a set of software tools and technologies to support the development process. The frontend of the application was built using React Native with the Expo CLI, while JavaScript served as the main programming language. The backend relied on Firebase services, including Firestore was used as the real-time database to store product details, cargo information, and load plans. Firebase Authentication handled secure user login and account management, while Firebase Storage was utilized to store product and cargo images.

For development tools, Visual Studio Code was chosen as the primary code editor, and Android Studio was used to configure the emulator and assist in debugging. Several supporting libraries were also included, such as Expo-camera for barcode scanning, Expo-linear-gradient, Ionicons, and FontAwesome5 for UI enhancements, and React Navigation to manage screen transitions and navigation flows effectively.

### 6.2.3 Configuration Setup

This project starts with creating a Firebase project on https://console.firebase.google.com/. The project included enabling Firestore Database for storing product, cargo, and load plan data, Firebase Authentication for user login and registration, and Firebase Storage for saving images. A configuration file (google-services.json) was generated and later added to the application folder.

The mobile app was set up using Expo CLI by running 'npx create-expo-app <project-name>'. Node.js and npm were installed to support package management, while Android Studio was configured to run the app using the Pixel 4a emulator (API 30). This allowed the application to be tested both on real devices and in the emulator.

The application required several libraries to provide its core functionalities, which were installed using npm. These included the Firebase SDK for Firestore, Authentication, and Storage integration, expo-camera for barcode scanning, and expo-linear-gradient for interface design. For navigation management, @react-navigation/native and its related packages were implemented, while Ionicons and FontAwesome5 were used to enhance the user interface with icons. Additional supporting libraries were also installed as needed to improve functionality and user experience.

A dedicated configuration file (firebase.js) was created in the project's source folder. The Firebase configuration details including apiKey, projectId, storageBucket, etc. were copied from the Firebase Console and pasted into this file. Firestore, Authentication, and Storage were initialized and exported, allowing the entire application to access Firebase services consistently. With the development environment fully set up and Firebase

successfully integrated, the system was ready to begin coding and implementing the main application features.

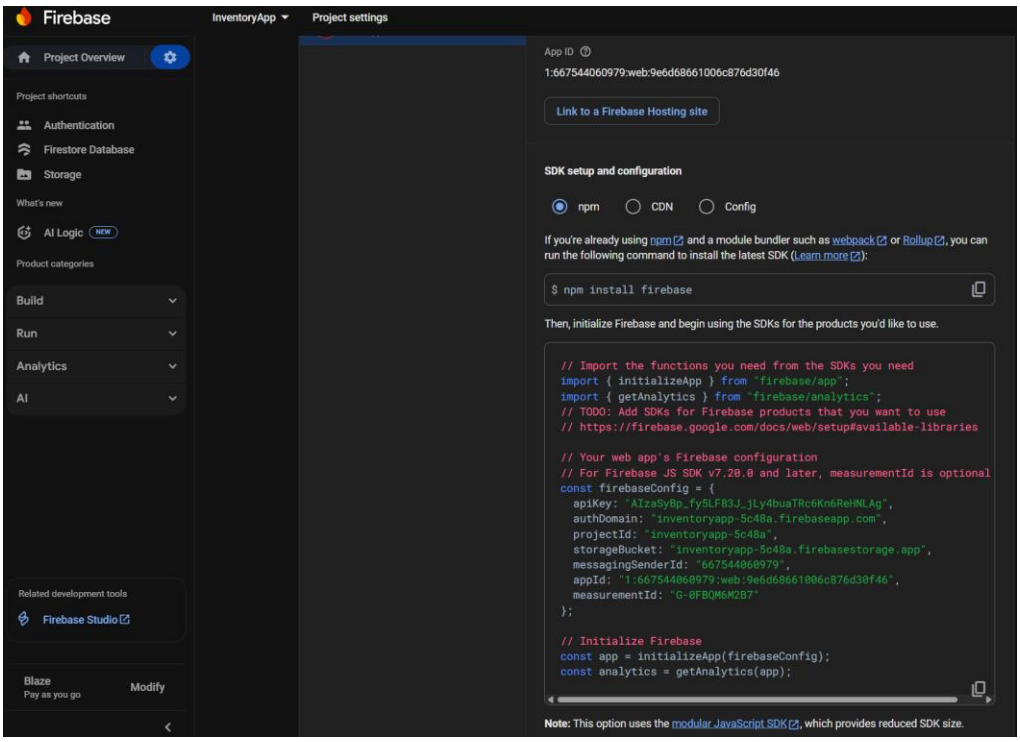



Figure 6. 1: Firebase configuration shown in Firebase console.

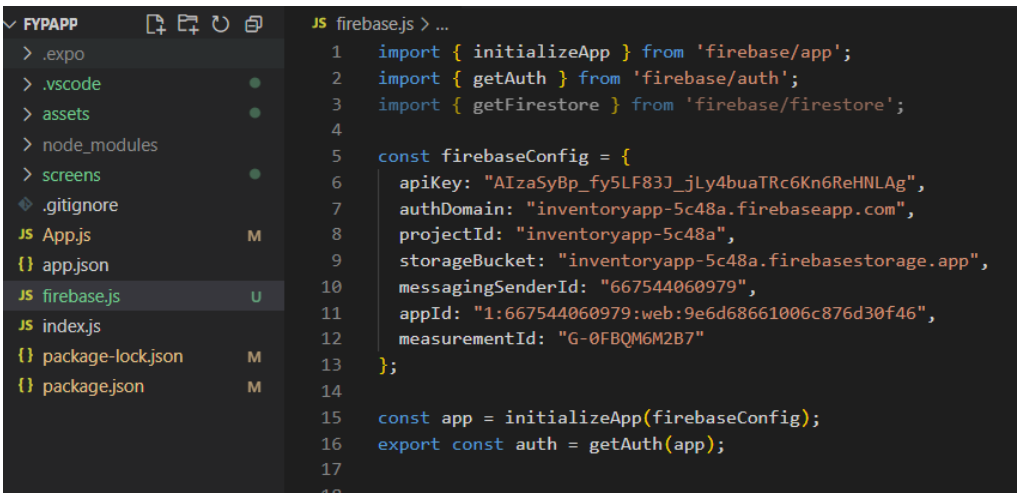



Figure 6. 2: Firebase configuration pasted in the Firebase config file.

## 6.3 System Modules

The implementation of the system was divided into three major modules which are User Management, Inventory Tracking, and Load Planning. Each module was developed according to the design specifications and integrated into the mobile application.

### 6.3.1 User Management Module

The User Management module is designed to handle account registration, login, and access control for the application. It was implemented using Firebase Authentication to provide secure account creation and login, while Firestore is used to store user-related information such as name and email. This module ensures that only authorized users can log in and access the system's features.

The first screen that appears when the user opens the application is the Welcome Screen, as shown in Figure 6.3. This screen introduces the application with a short description of its purpose, letting users know they are entering an inventory tracking system with barcode scanning and automated load planning. It provides a simple entry point with a Login button, which directs users to the authentication process managed by the User Management module.


Figure 6. 3: Welcome Screen.

The Login Screen allows users to access the system by entering their email and password. As shown in Figure 6.4, the system validates the input and will prompt an error message when the credentials are incomplete or incorrect. For example, if the user enters only the email without a password, the system displays a message indicating that both email and password field is required. Similarly, if the user enters an invalid email format, such as missing the proper Gmail structure, the system alerts the user with an error message.

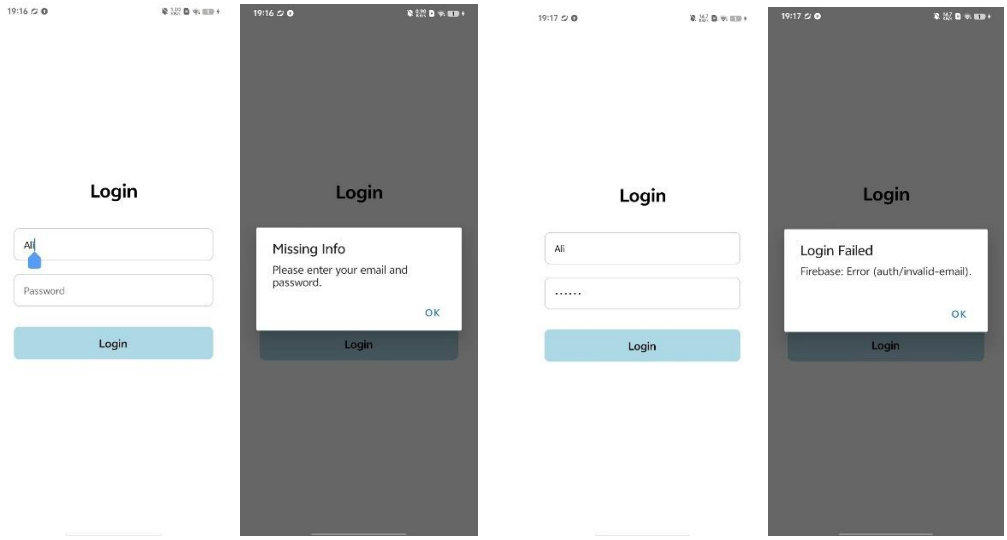



Figure 6. 4: Login Screen with Input Validation and Error Messages.

Only when both the email and password are provided in the correct format will the login be successful, and the user will be redirected to the home screen.


Figure 6. 5: Login Screen with Correct Email Format and Password.

After a successful login, the user is redirected to the Home Screen, which serves as the main dashboard of the application. This screen provides a quick overview of inventory status, including the number of products added, total products available, items restocked, and low stock alerts. A section at the bottom highlights items that have reached or fallen below the minimum stock level, allowing users to take immediate action. From the Home Screen, users can also navigate to other key features of the system such as product management, barcode scanning, load planning, and personal profile through the bottom navigation bar.

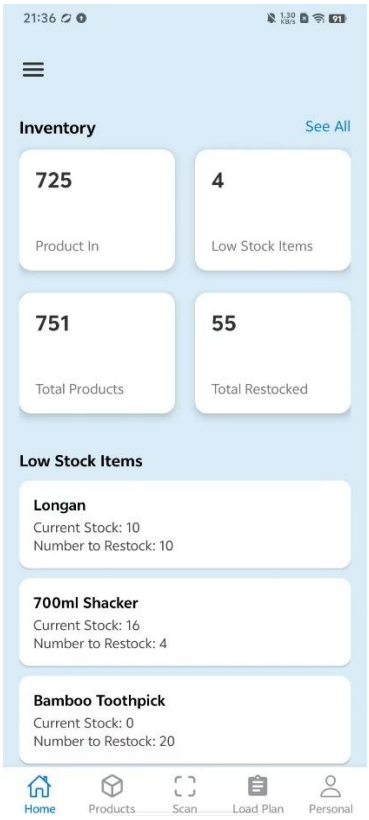



Figure 6. 6: Home Screen.

When the user navigates to the Personal tab, the system displays the logged-in user's profile information, including their name and unique ID. This screen also provides a button to access User Management, where all registered accounts can be viewed. A logout option is also available, allowing users to securely exit the application.


Figure 6. 7: Personal Screen.

The User Management screen shows a list of all registered users in the system. Each record includes the username, email address, and unique user ID, which are managed through Firebase Authentication and Firestore. This feature allows users to view existing accounts stored in the database.

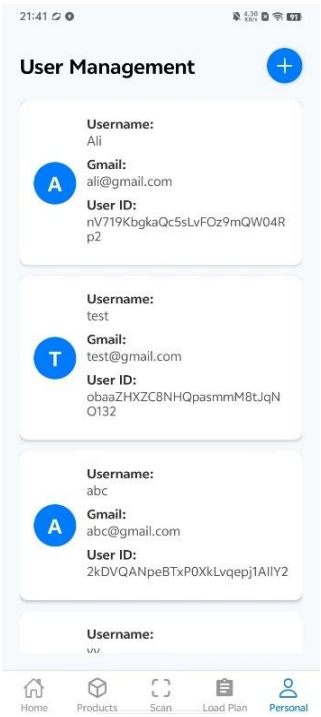



Figure 6. 8: User Management Screen.

The Add New User form enables the creation of new accounts by entering a name, email, and password. Once submitted, the system processes the registration using Firebase Authentication, while Firestore stores the additional details for reference. This provides a simple way to expand the list of users who can log in to the application.

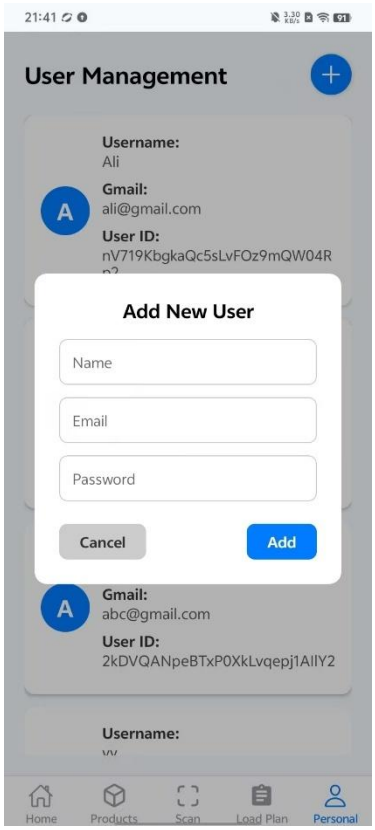



Figure 6. 9: Add New User Screen.

### 6.3.2 Inventory Tracking via Barcode Scanning Module

The Inventory List displays all products stored in the system, including their name, ID, category, image, available stock quantity, minimum stock quantity and stock status. Users can easily browse the list, while items with low or zero stock are highlighted with a red indicator for quick identification. The product information is retrieved in real time from Firebase Firestore, ensuring that any changes made during stock in or out updates or barcode scanning are immediately reflected in the list.

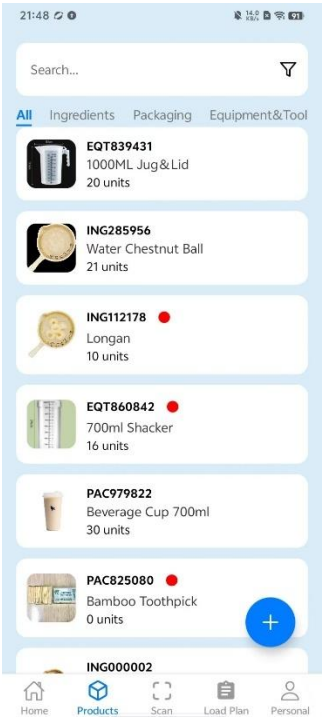



Figure 6. 10: Inventory List Screen.

A search bar and category filters are provided at the top of the Inventory List to make it easier for users to find specific products. The search bar allows users to type in a product ID or name, and the system will instantly display matching results from the Firestore database. This reduces the time spent scrolling through long lists of products.

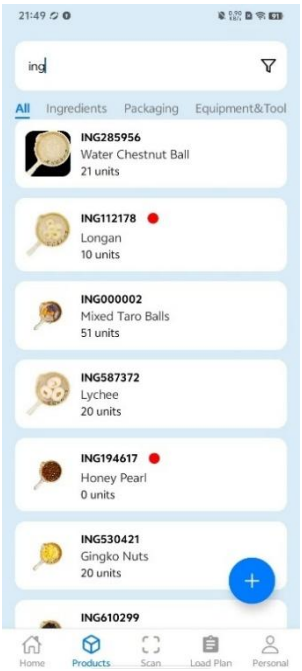



Figure 6. 11: Search Inventory List Using Product ID Prefix.

In addition, the category filters such as Ingredients, Packaging, and Equipment allow users to narrow down the list based on product type. For example, selecting ‘Ingredients’ will only display items categorized under that group, hiding unrelated products.

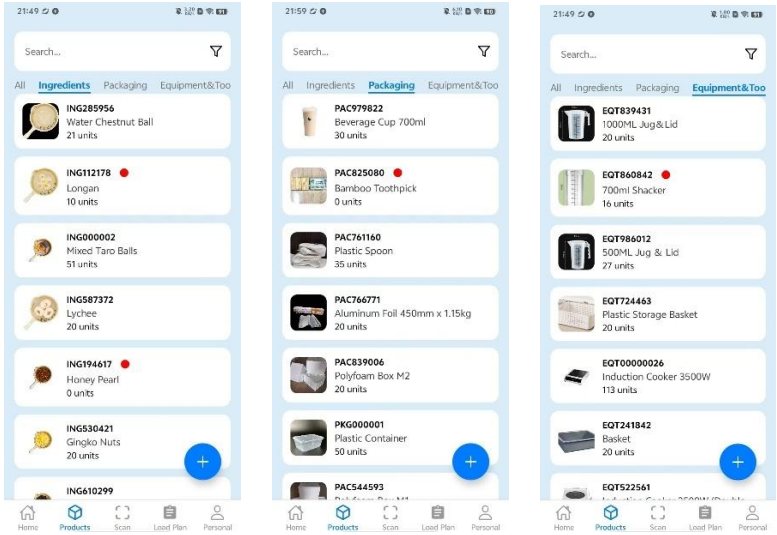



Figure 6. 12: Inventory List with Category Filters.

In addition, the filter function allows users to narrow down the product list based on stock status. Selecting In Stock displays only items that currently have available quantity, while Out of Stock shows products with zero units remaining. The Low Stock option highlights products that have reached or fallen below their predefined minimum stock level. These filters help users quickly focus on items that require restocking or monitoring.

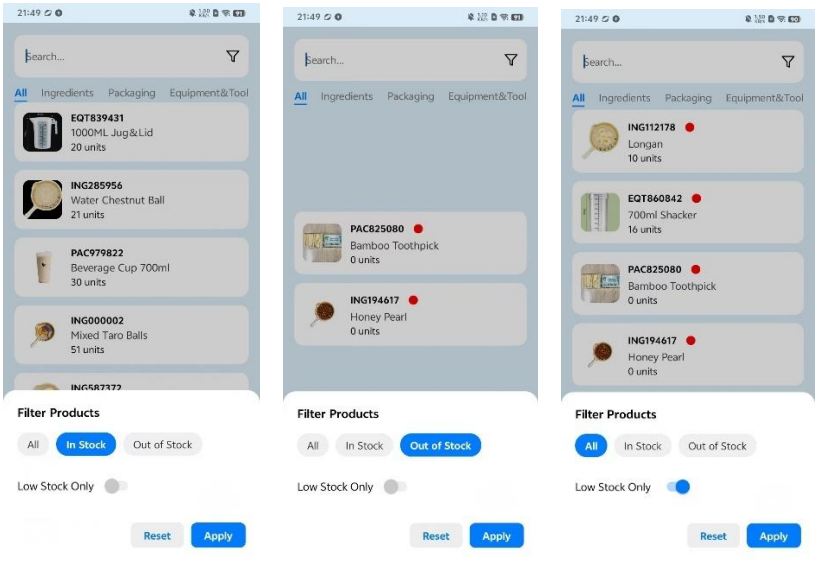



Figure 6. 13: Inventory Filter Options.

Next, as shown in Figure 6.14, user is allowed to create a new inventory item by entering the product name, quantity, category and minimum stock quantity. An image can be attached by tapping the "+" placeholder. When Save is pressed, the record is created in Firebase Firestore and immediately appears in the Inventory List.

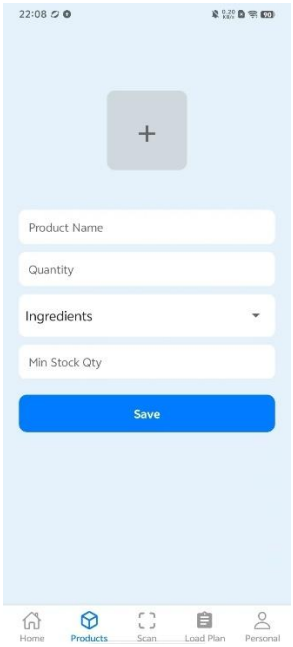



Figure 6. 14: Add Product Form.

When adding an image, the app requests permission and opens the device gallery. After the user selects a photo, it is uploaded to Firebase Storage, and the product document stores the image URL in Firestore.


Figure 6. 15: Select Image for Product from Gallery.

When a user taps on a product card in the Inventory List, the system retrieves and displays the complete product details from Firebase Firestore. This includes the product ID, name, category, available quantity, minimum stock quantity, and current stock status (e.g., In Stock, Low Stock, or Out of Stock).

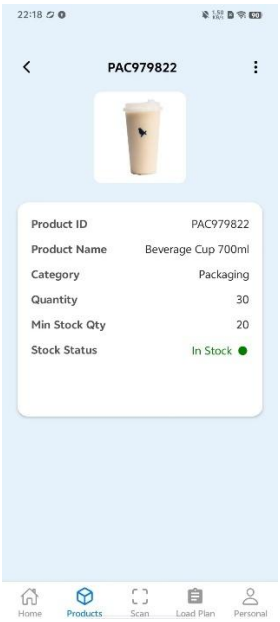



Figure 6. 16: Product Detail Screen.

From Product Details screen, the user can also choose to update product details. By clicking the ‘Edit’ button and switching into edit mode, fields such as name and category can be modified. After pressing Save, the updated values are written back to Firestore, and the changes are immediately reflected in the inventory list.

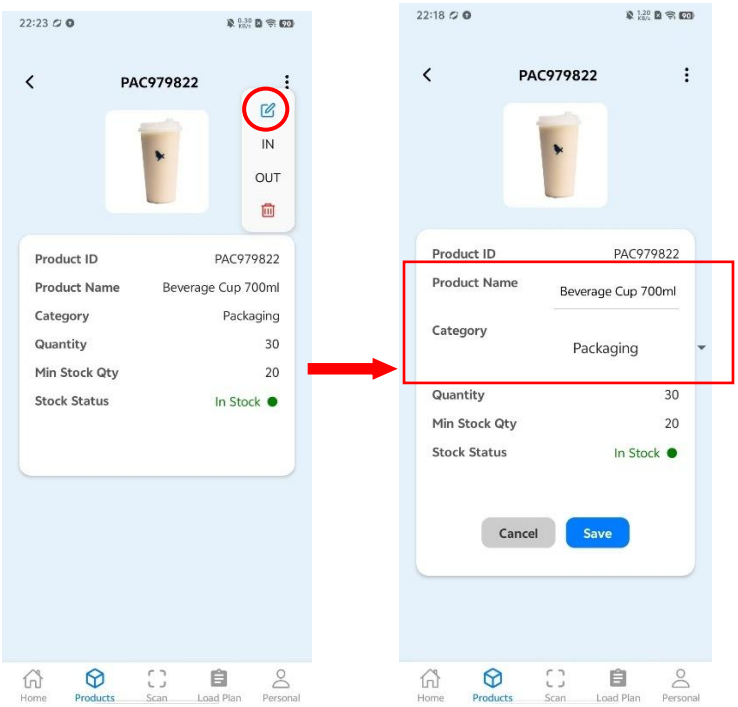



Figure 6. 17: Product Detail Screen in Edit Mode.

The system also provides the option to delete products that are no longer in use. When a user selects the delete option from the product detail screen, a confirmation dialog appears to prevent accidental deletion. Only after the user confirms by pressing Delete will the product be permanently removed from Firebase Firestore. If the user cancels, no changes are made, and the record remains intact.

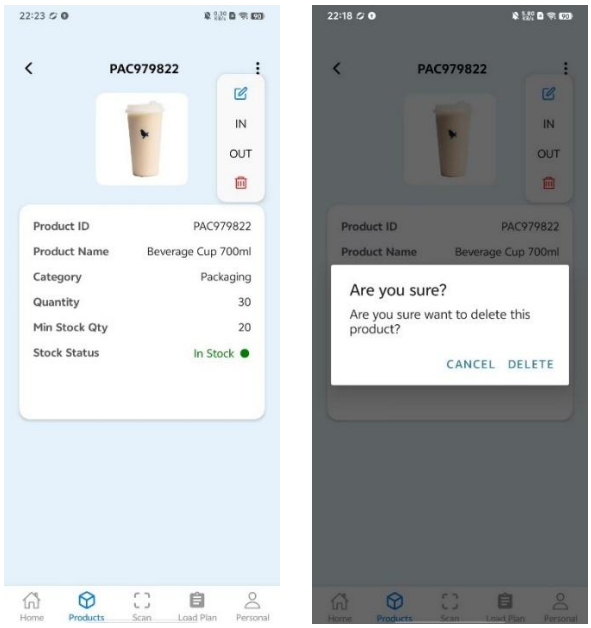



Figure 6. 18: Product Deletion with Confirmation Dialog.

The system also provides stock in or out functions directly from the product detail screen, mainly used when updating the quantity of a single product. When the user selects In, a dialog appears to enter the quantity to be added to stock, while choosing Out allows the user to specify the quantity to be deducted. Stock levels are updated accurately and in real time within the Firestore database. The system then automatically compares the updated stock with the minimum stock quantity. If the quantity falls below the threshold, the status changes to Low Stock, highlighted in red for immediate attention. Conversely, when stock is replenished above the threshold, the status updates to In Stock, marked in green. This feature helps maintain effective inventory control and prevents stock shortages.

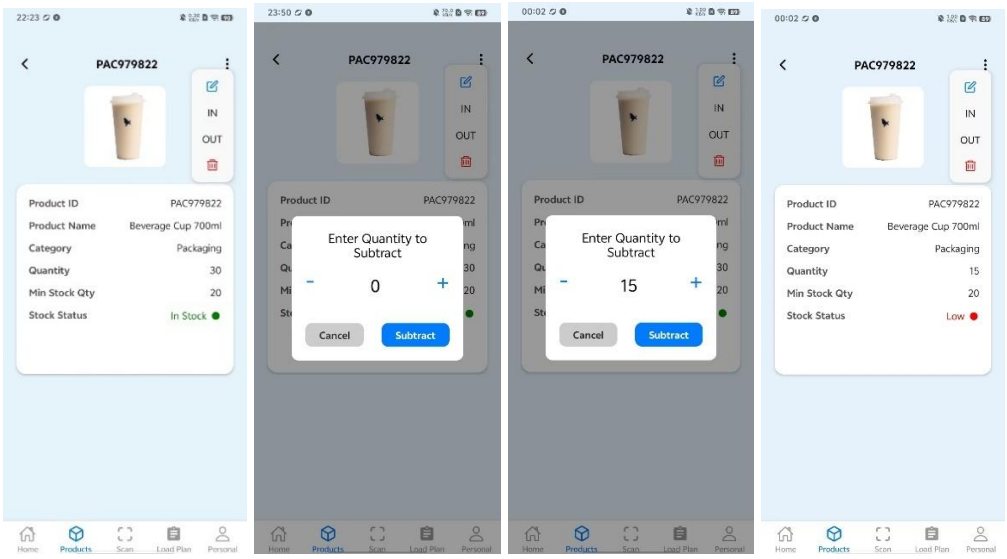



Figure 6. 19: Stock Updated to Low After Product Out.

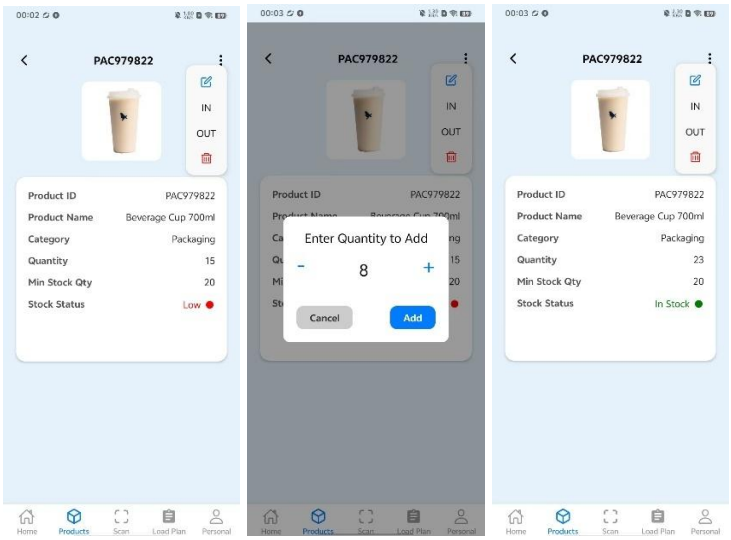
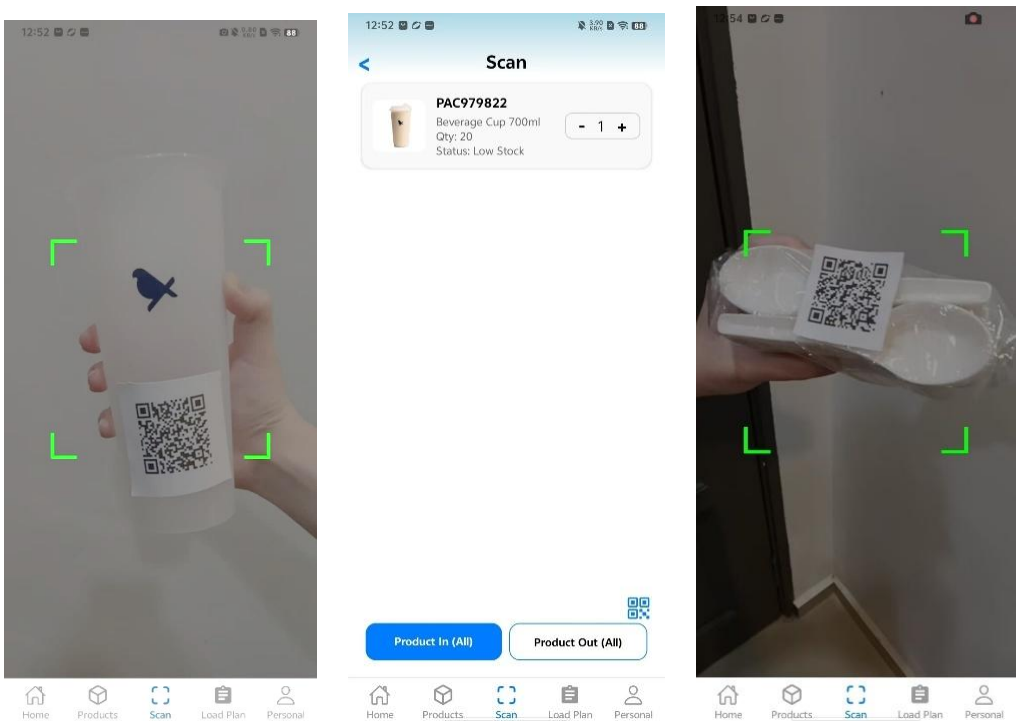



Figure 6. 20: Stock Restored to In Stock After Product In.

The Scan function provides a faster way to manage stock quantities for multiple products at once. Users can simply scan the QR codes attached to product packaging, and the system will automatically retrieve product details using product id from the Firestore database. Once scanned, the products appear in a list with their current quantities and stock status. From here, users can adjust the quantity by increasing or decreasing the values for each product. This feature is especially useful when handling bulk stock updates, such as during goods receipt or dispatch, since it allows users to update several products in a single session.

After adjustments, users can confirm by selecting either Product In (All) or Product Out (All), which updates the stock levels of all scanned products in real time. The system also validates against the minimum stock quantity, updating each product's status accordingly.



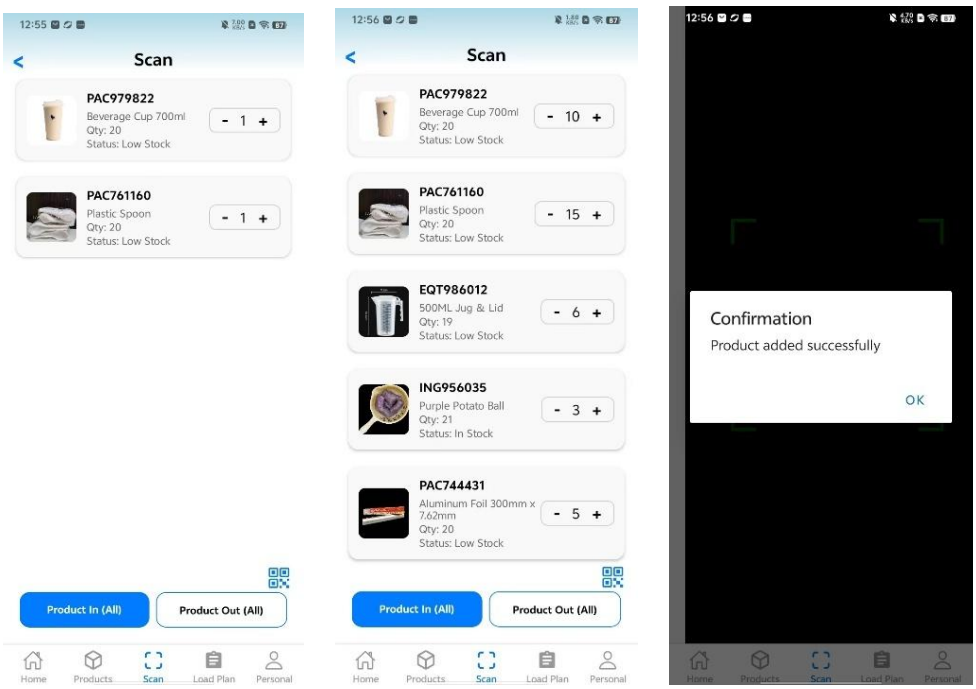



Figure 6. 21: Barcode Scanning for Bulk Product In/Out Updates.

### 6.3.3 Load Planning Module

When users navigate to the Load Plan module, they are presented with three main options which are Plan Load, Set Common Size, and History. This serves as the entry point for creating new load arrangements, defining frequently used sizes, or reviewing past load plans.

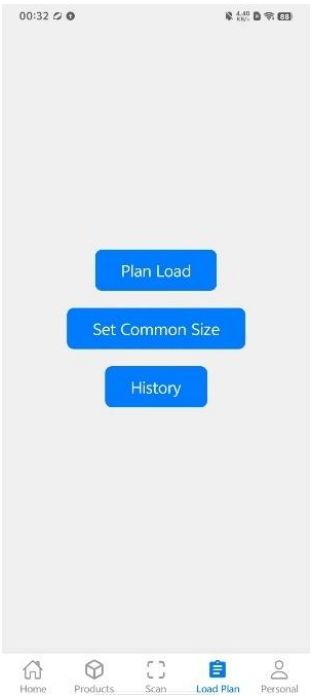



Figure 6. 22: Load Plan Main Menu.

In the screen shown in Figure 6.23, users can register a new container by providing details such as container name, dimensions (length × width × height), and maximum load weight. Alongside container setup, users can add cargo items by entering cargo name, dimensions, and weight.


Figure 6. 23: Add New Container and Cargo Screen.

In the Plan Load screen, users can begin by selecting both a container and cargo items. The interface provides separate sections for ‘Container’ and ‘Cargo’ with a Confirm Selection button to proceed once the choices are made. The system displays all available containers saved earlier. Users can choose from multiple options, depending on the vehicle or container type required for the load plan. Once a container is chosen, its details such as dimensions and maximum load capacity are displayed for confirmation.

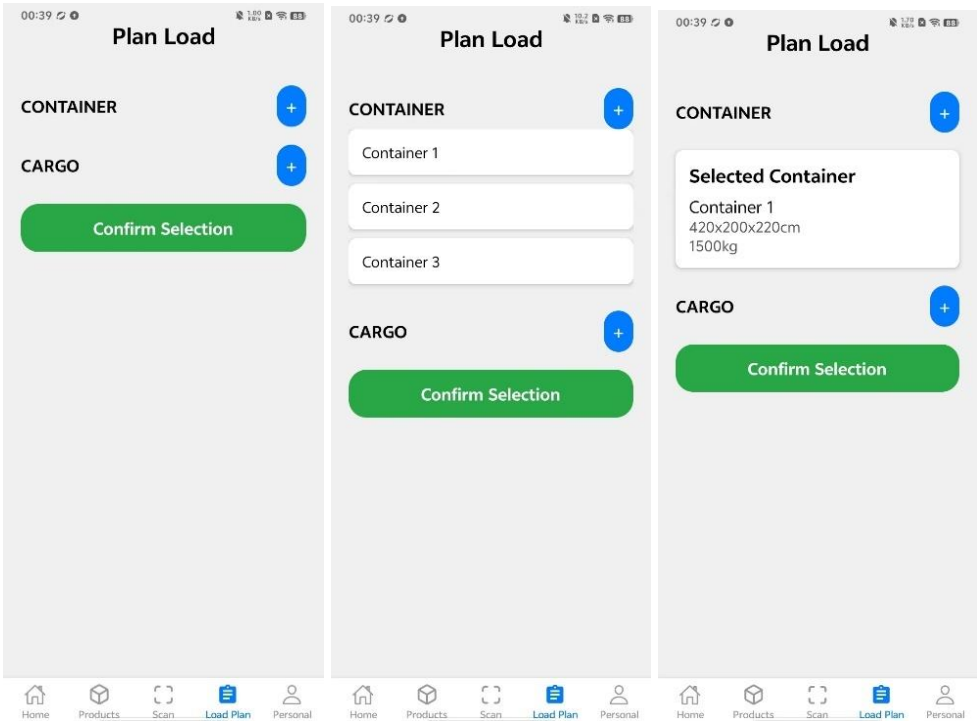



Figure 6. 24: Selecting Container from List.

The user can then select different cargo items to include in the load plan. Each cargo is displayed with its dimensions and weight, allowing users to make informed decisions on what to load. After selecting cargo items, the system allows users to adjust the quantity for each type of cargo.

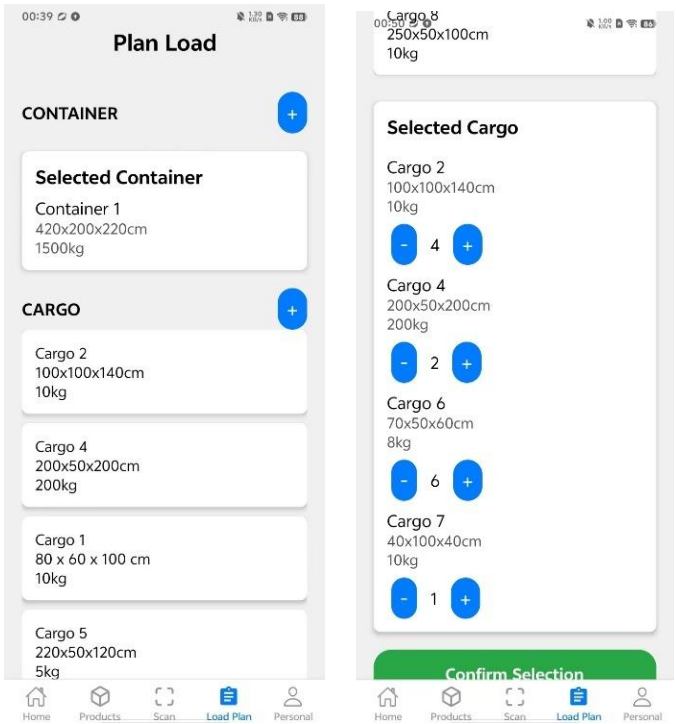



Figure 6. 25: Adding Cargo and Adjusting Cargo Quantities.

If the total cargo weight goes beyond the selected container's maximum limit, the system issues a warning message 'Weight Limit Exceeded'. This prevents unsafe or invalid load plans and ensures compliance with container restrictions.

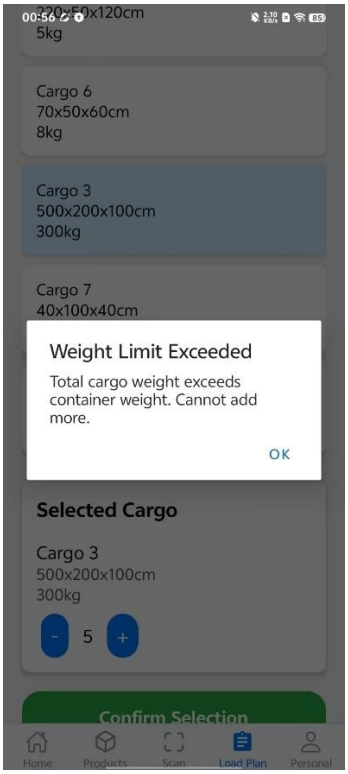



Figure 6. 26: Exceeding Container Weight Limit.

After selecting the desired container and cargo, the user clicks the 'Confirm Selection' button to proceed. At this stage, the system automatically arranges the selected cargo within the container using the load planning algorithm. As shown in Figure 6.27, the system generates an Actual Ratio Diagram that visually represents the arrangement of the cargo. Each cargo is both color-coded and labelled with text, ensuring clear distinction between different items. The inclusion of text labels also makes the diagram understandable even when printed in black and white, so users are not restricted when downloading or sharing the load plan in different formats.

In addition, the system calculates and displays space utilization details, including the total container space, used space, and free space. This information helps the user evaluate how efficiently the cargo has been packed. If certain cargo cannot be fitted into the container, the system highlights them

under the ‘Not Fit into Container’ section. This ensures users are aware of unplaced items immediately and can make adjustments such as selecting a larger container or splitting cargo into multiple containers.

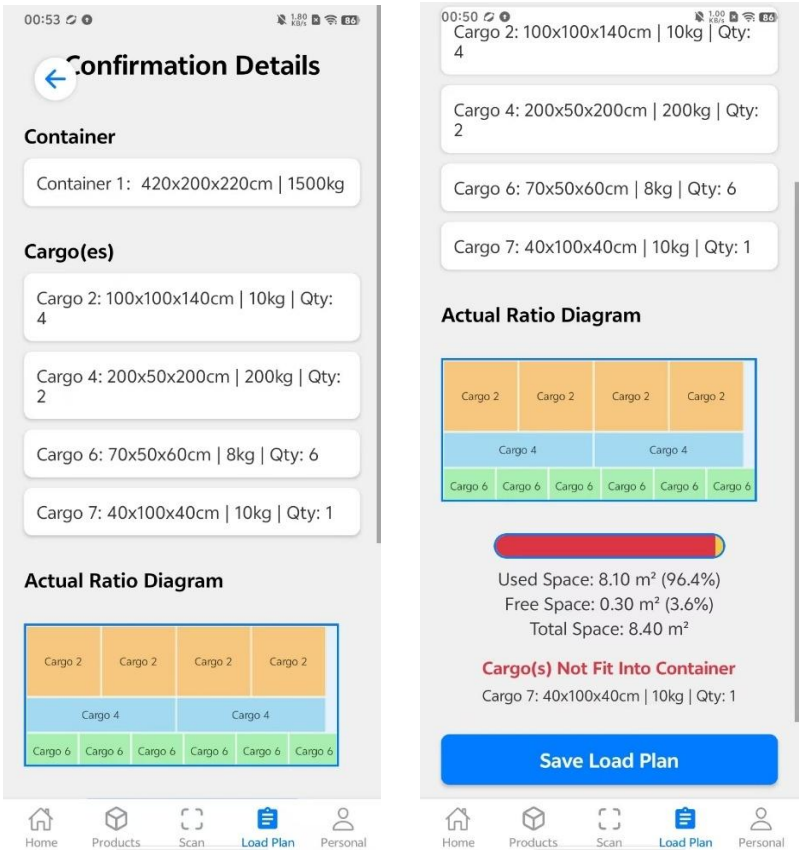



Figure 6. 27: Automatic Cargo Arrangement.

Besides the automatic arrangement, the system also allows users to manually adjust cargo placement through a drag-and-drop interface as shown in Figure 6.28. This feature allows users to manually adjust cargo positions if they prefer customized arrangement instead of the automatically generated one.

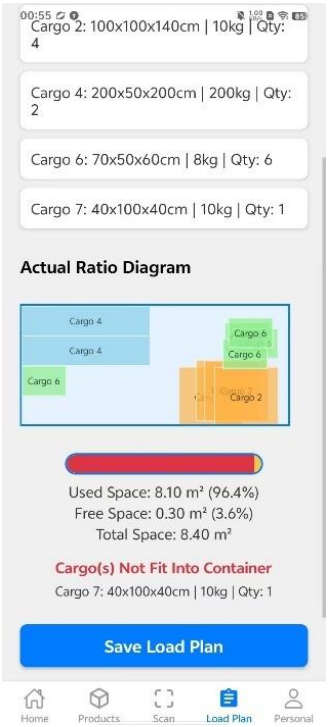



Figure 6. 28: Manual Adjustment of Cargo Placement Using Drag-and-Drop.

After the load plan is saved, the system automatically records it in the Load Plan History, as shown in Figure 6.29. Each saved entry is listed with a unique ID, container details, total used and free space, the list of cargoes included, and the date and time the plan was created.

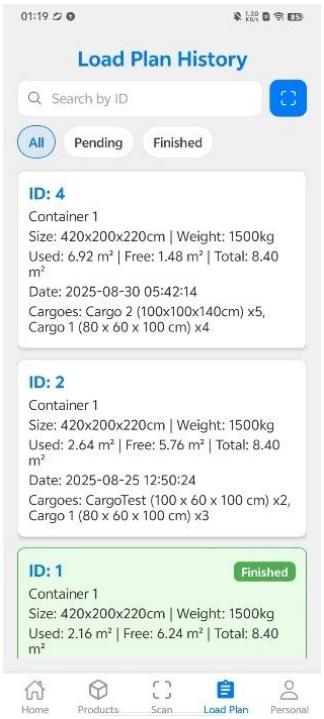



Figure 6. 29: Load Plan History screen.

Users can filter the history by All, Pending, or Finished status to quickly locate specific load plans. A search bar is also available for searching directly by load plan ID.

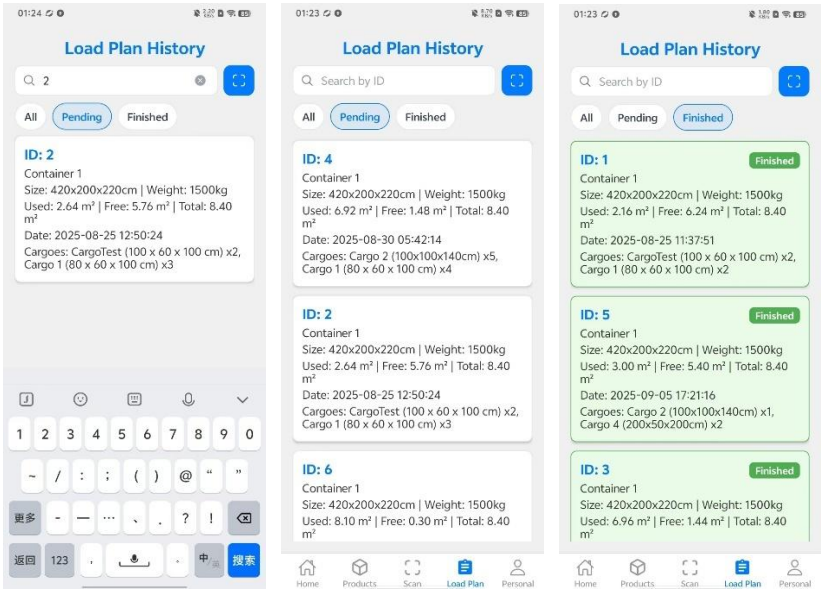



Figure 6. 30: Load Plan Shown with Filters and Search Option.

Selecting a particular history record displays the full details of that plan, as illustrated in Figure 6.31. The screen shows the container used, cargo details with their dimensions, weight, and quantity, along with a color-coded arrangement diagram. Below the diagram, the used space and free space values are shown.

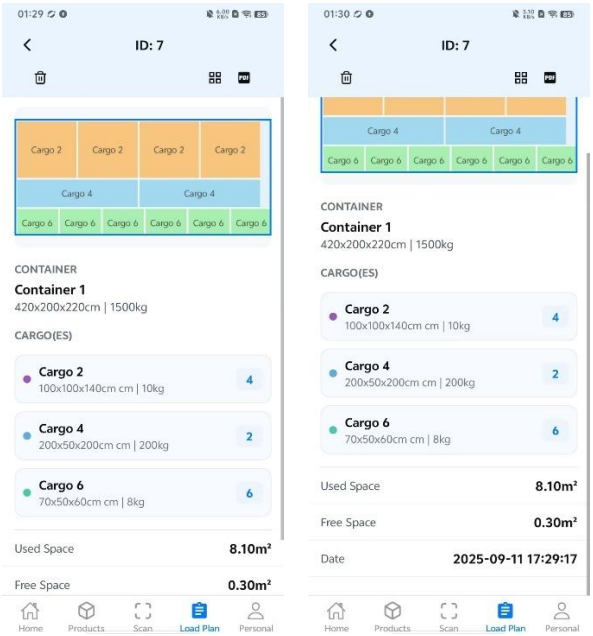



Figure 6. 31: Detailed View of a Selected Load Plan.

After saving a load plan, the system provides the option to generate a PDF report by clicking the PDF button. The PDF can be shared, downloaded, or printed directly from the application for documentation or operational use. This feature ensures that load plans can be easily distributed to logistics staff, stored for record-keeping, or printed for on-site reference.

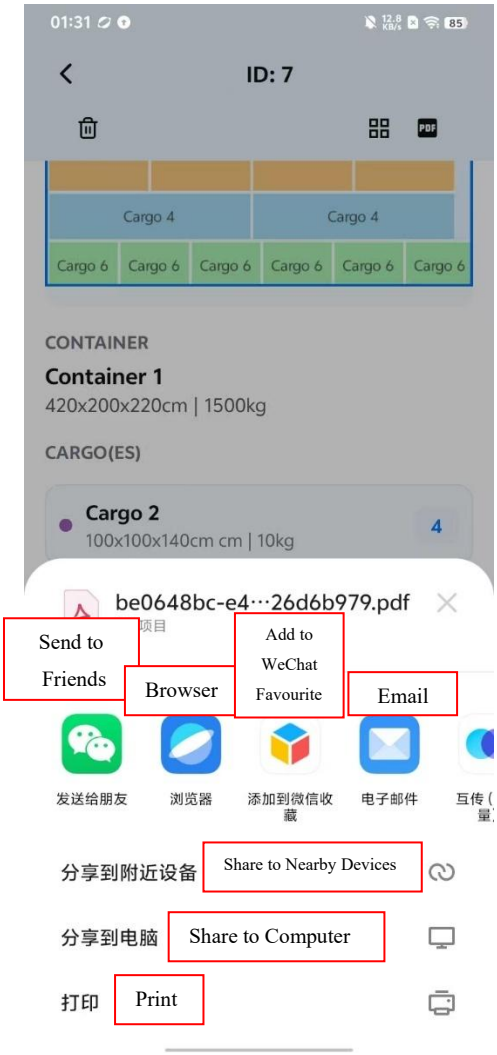



Figure 6. 32: Options to Share, Download, or Print the Load Plan.

As shown in Figure 6.33, the PDF report includes container specifications, cargo details, total used and free space, cargo arrangement order, and a visual diagram of the actual placement. Additionally, a QR code is embedded in the PDF, allowing users to quickly retrieve and view the same load plan details in the system by scanning it.

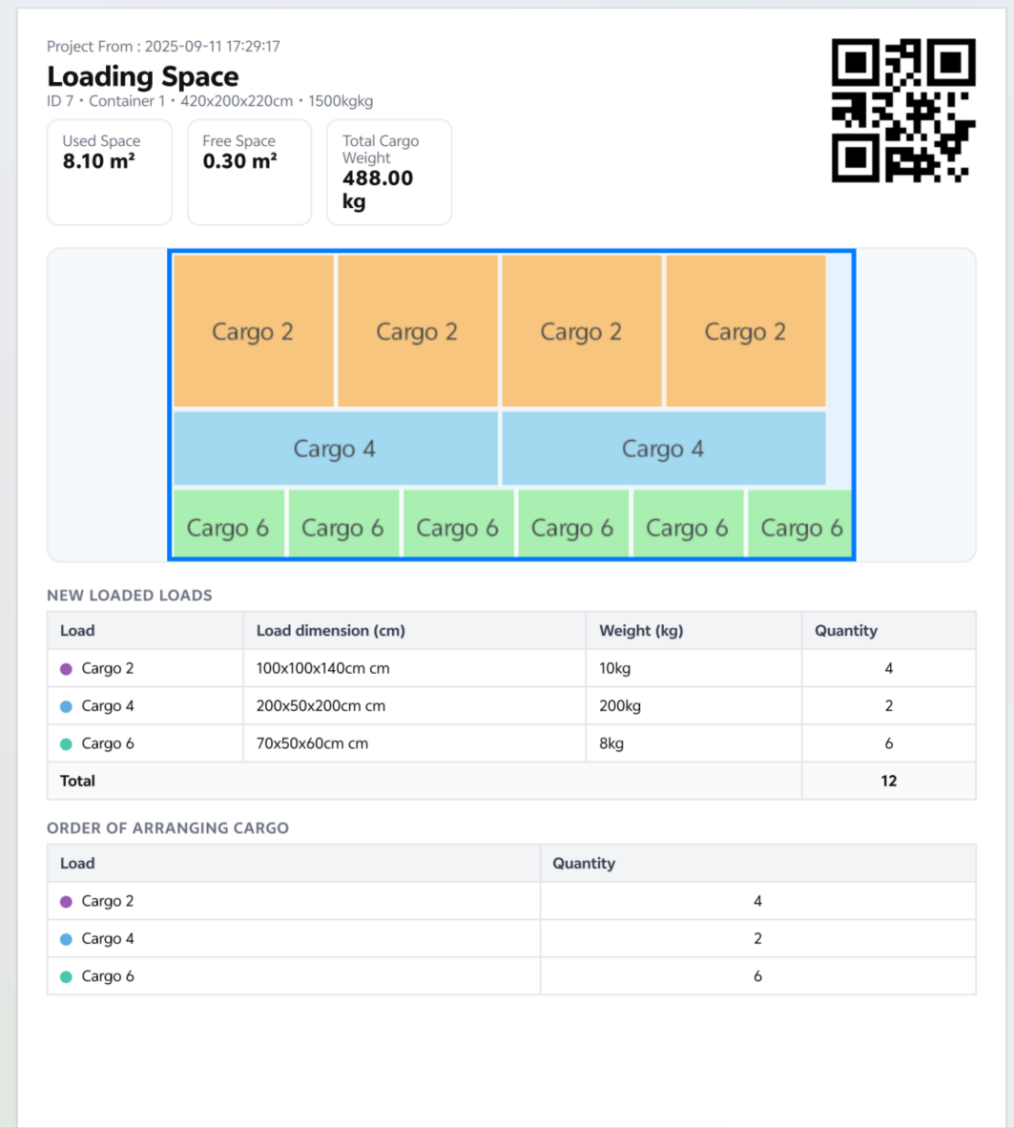



Figure 6. 33: Generated PDF report.

After generating and saving the load plan, users can retrieve it by scanning the QR code printed on the PDF. When the QR code is scanned, the corresponding load plan details are automatically displayed in the app.

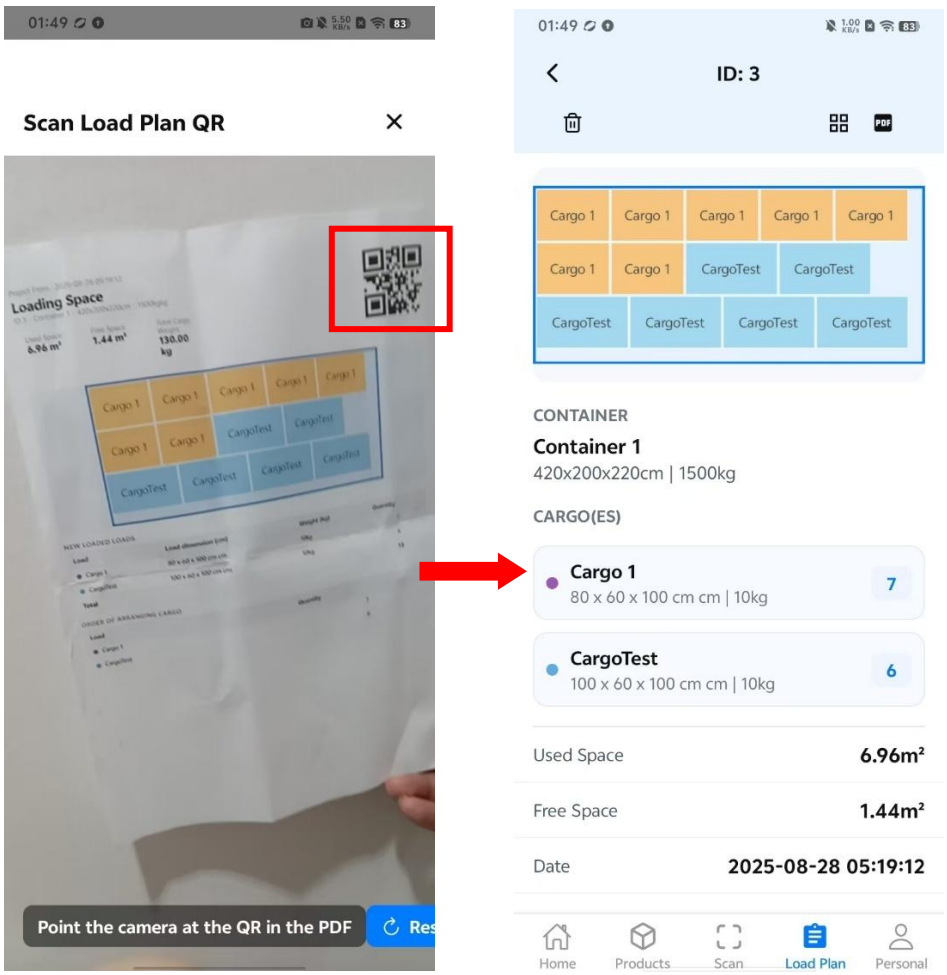



Figure 6. 34: Scanning the QR code from the printed load plan PDF.

The system then provides a Cargo Arranging Checklist, as shown in Figure 6.35, which lists all the cargo items that need to be placed into the container. This checklist guides the user step by step in following the planned arrangement generated earlier by the system, ensuring that no cargo item is missed out during the physical loading process. Each item can be marked as 'Arranged' once it has been physically placed inside the container. After all items have been marked as arranged, the user finalizes the process by clicking the 'Save (Settled)' button. At this stage, the status of the load plan is automatically updated from Pending to Finished, ensuring that the plan is properly tracked and updated in the system.

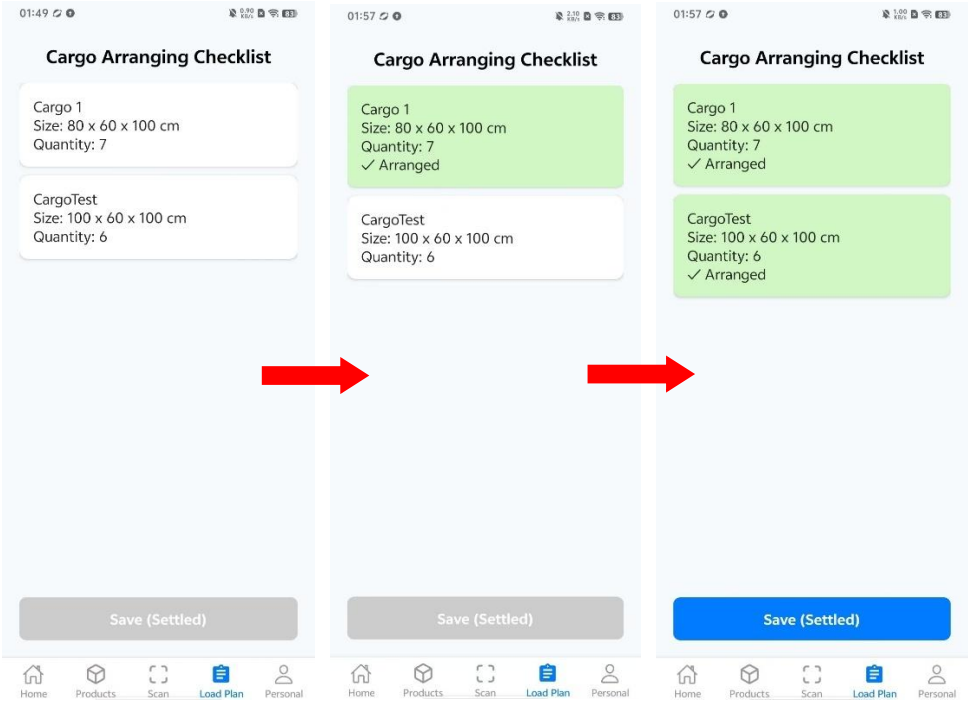



Figure 6. 35: Cargo arranging checklist.

The system also provides a delete function. When the user chooses to delete a load plan, the system displays a confirmation dialog, as shown in Figure 6.36. This dialog ensures that the user does not accidentally remove an important record. The dialog presents two options which is Cancel, which aborts the action and retains the load plan, or Delete, which permanently removes the selected plan from the system.

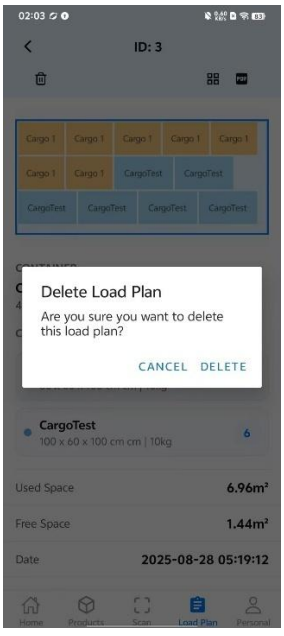



Figure 6. 36: Confirmation dialog for deleting a load plan.

# CHAPTER 7

# SYSTEM TESTING

## 7.1 Introduction

In this project, system testing is conducted to evaluate the functionality, usability, and overall quality of the developed application. This chapter presents and discusses the testing approaches carried out, including unit testing to verify individual components, integration testing to ensure proper interaction between modules, the System Usability Scale (SUS) test to assess user experience, and the User Acceptance Test (UAT) to ensure that the system meets user needs and project objectives.

## 7.2 Unit Testing

Unit testing focuses on testing individual components or modules of the system to ensure they function correctly in isolation. In this project, a total of 22 test cases were executed, covering all three main modules of the application. Each of these test cases was further expanded into detailed scenarios, resulting in a total of 91 sub-test cases. This approach ensured that both normal and exceptional conditions were thoroughly tested, covering functional correctness, data handling, and user interactions.

### 7.2.1 Unit Test Cases Listing

Table 7. 1: Summary of Unit Test Cases Listing.

| Test Module | Test Case ID | Test Case Title | Status |
|---|---|---|---|
| User Management | TC01 | Add new user | Pass |
| | TC02 | Login account | Pass |
| | TC03 | Logout | Pass |
| Inventory Tracking | TC04 | Home Dashboard | Pass |
| | TC05 | View Products List | Pass |
| | TC06 | Add new product | Pass |
| | TC07 | View Product Detail | Pass |
| | TC08 | Edit Product Info | Pass |
| | TC09 | Stock Update using Product In | Pass |
| | TC10 | Stock Update using Product Out | Pass |
| | TC11 | Delete Product | Pass |
| | TC12 | Camera Permission and Preview | Pass |
| | TC13 | Scan the QR code on inventory item | Pass |
| Load Planning | TC14 | Navigate from Load Plan home | Pass |
| | TC15 | Container and cargo selection | Pass |
| | TC16 | Confirmation Details and Actual Ratio Diagram | Pass |
| | TC17 | Generate and save load plan | Pass |
| | TC18 | View load plan history | Pass |
| | TC19 | View load plan detail | Pass |
| | TC20 | Arrange cargo checklist | Pass |
| | TC21 | Set common size for containers and cargo | Pass |
| | TC22 | Export a plan to PDF | Pass |

### 7.2.2 Unit Test Cases

#### 7.2.2.1 User Management Module

Table 7. 2: Unit Test Case of Add New User.

| Test Case Title | Add new user | | Test Module | User Management | |
|---|---|---|---|---|---|
| Test Case ID | TC01 | | | | |
| Pre-conditions | The application is launched and the user is logged in. Navigate to User Management screen. "Add New User" modal is opened after tapping the **"+"** button. | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT01 | Add user with valid name, email, password | 1. Enter valid name, email, and password.<br>2. Tap Add. | Name: abc<br>Email: abc@gmail.com<br>Password: Abc12345 | New user created and info stored in Firestore.<br>Success alert shown.<br>User displayed in list. | Pass |
| UT02 | Add new user with empty name | 1. Leave Name field blank.<br>2. Enter valid email and password.<br>3. Tap Add. | Name: " "<br>Email: empty@gmail.com<br>Password: Abc12345 | Alert "Please fill all fields." shown. No user created. | Pass |

| | | | | | |
|---|---|---|---|---|---|
| UT03 | Add new user with Invalid email format | 1. Enter valid name and password.<br>2. Enter invalid email.<br>3. Tap Add. | Name: yanxin<br>Email: yanxin@<br>Password: yanxin77777 | Error alert displayed for invalid email. User not created. | Pass |

Table 7. 3: Unit Test of Login account.

| Test Case Title | Login account | | Test Module | User Management | |
|---|---|---|---|---|---|
| Test Case ID | TC02 | | | | |
| Pre-conditions | User is on the Login Screen. At least one user account exists in Firebase Authentication. | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT04 | Login with predefined (valid) info | 1. Enter valid email and password created by admin.<br>2. Tap Login. | Email: abc@gmail.com<br>Password: Abc12345 | Successfully logged in and navigates to Home screen. | Pass |
| UT05 | Login with not predefined (unregistered) info | 1. Enter email not in system and password.<br>2. Tap Login. | Email: nono@gmail.com<br>Password: Abc12345 | Error alert “Invalid credentials” shown. Stay on Login screen. | Pass |
| UT06 | Login with invalid | 1. Enter valid registered | Email: abc@gmail.com | Error alert “Invalid | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | password | email.<br>2. Enter wrong password.<br>3. Tap Login. | Password: wrong123 | credentials" shown. Stay on Login screen. | |
| UT07 | Login with empty fields | 1. Leave email and password blank.<br>2. Tap Login. | Email: " "<br>Password: " " | Alert "Please enter your email and password" shown. No login attempt made. | Pass |
| UT08 | After login, Personal screen shows correct Name and ID | 1. Login with valid credentials.<br>2. Navigate to Personal tab. | Email: abc@gmail.com<br>Password: Abc12345 | Personal screen displays name, abc, ID and Avatar shows first letter A. | Pass |

Table 7. 4: Unit Test Case of Logout.

| Test Case Title | Logout | | Test Module | User Management | |
|---|---|---|---|---|---|
| Test Case ID | TC03 | | | | |
| Pre-conditions | User is already logged in successfully and navigates to the Personal tab screen. | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT09 | Successful logout | 1. Tap Logout icon. | - | User session cleared. | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | | | App navigates back to Welcome Screen. | |
| UT10 | Attempt to access app after logout | 1. Log out successfully.<br>2. Relaunch the app or navigate to protected screen. | - | App forces user to Login Screen. User cannot access features until login again. | Pass |

#### 7.2.2.2 Inventory Tracking via Barcode Scanning Module

Table 7. 5: Unit Test Case of Home Dashboard.

| Test Case Title | Home Dashboard – Hamburger Menu & Navigation | | Test Module | Inventory Tracking | |
|---|---|---|---|---|---|
| Test Case ID | TC04 | | | | |
| Pre-conditions | User is on Home screen. Hamburger icon visible at top-left. | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT11 | Inventory analysis info is displayed | 1. Open Home screen.<br>2. Wait for Firestore data to load. | Products in Firestore with different quantity, stockStatus, minStockQty. | All 4 cards display correct values. The Low Stock section lists | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | | | items with current stock & restock numbers. | |
| UT12 | Open menu sheet | 1. Tap hamburger menu icon. | - | Side sheet opens. | Pass |
| UT13 | Close by tapping outside | 1. Tap hamburger menu icon.<br>2. Tap outside overlay. | - | Menu closes. | Pass |
| UT14 | Navigate to Products | 1. Tap hamburger menu icon.<br>2. Tap ‘Products’. | - | Menu closes; navigates to Products screen. | Pass |
| UT15 | Navigate to Load Plan | 1. Tap hamburger menu icon.<br>2. Tap ‘Load Plan’. | - | Menu closes; navigates to Load Plan screen. | Pass |

Table 7. 6: Unit Test Case of View Products List.

| Test Case Title | View Products List | Test Module | Inventory Tracking |
|---|---|---|---|

| Test Case ID | TC05 | | | | |
|---|---|---|---|---|---|
| Pre-conditions | User logged in and on Products screen. Firestore products table exists. | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT16 | Display all product in list | 1. Open Products screen | Products: ING285956, EQT860842, PAC825080 | All products are listed with image, productId, name, and quantity. | Pass |
| UT17 | Search by product name | 1. Enter “Longan” in search bar | Name = “Longan” | Only “Longan” item appears in the list. | Pass |
| UT18 | Search by product ID | 1. Enter “EQT839431” in search bar | ID = “EQT839431” | Only “1000ML Jug&Lid” item appears. | Pass |
| UT19 | Filter by category | 1. Tap Packaging tab | Products with category Packaging | Only Packaging products (PAC…) displayed. | Pass |
| UT20 | Filter by stock status | 1. Tap filter icon<br>2. Select Out of Stock | ING194617 (qty=0) | List shows only out-of-stock items. | Pass |
| UT21 | View Low Stock Products | 1. View product list | EQT860842 qty=16, min=20 | Red dot shown beside low stock products. | Pass |
| UT22 | View Product Detail | 1. Tap any product card | PAC825080 | Show correct product | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | | | info. | |

Table 7. 7: Unit Test Case of Add New Product.

| Test Case Title | Add new product | | Test Module | Inventory Tracking | |
|---|---|---|---|---|---|
| Test Case ID | TC06 | | | | |
| Pre-conditions | Taps '+' floating button on product list screen to open Add Product form. | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT23 | Add product with all valid fields | 1. Tap image box and select product image from gallery.<br>2. Enter valid Product Name, Quantity, Category, Min Stock Qty.<br>3. Tap Save button. | Name: "Honey Pearl"<br>Qty: 30<br>Category: Ingredients<br>Min: 20 | New product added with success alert shown. | Pass |
| UT24 | Add product with missing fields | 1. Leave Product Name blank..<br>2. Tap Save button. | Name: " "<br>Qty: 20<br>Category: Ingredients | Alert popup displayed with message "Please fill in all fields." Stay | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | | Min Stock Qty: 10 | on Add Product screen. | |
| UT25 | Add product without image | 1. Do not select any image.<br>2. Fill in all other valid fields.<br>3. Tap Save button. | Name: Bamboo Straw<br>Qty: 15<br>Category: Packaging<br>Min Stock Qty: 30 | Product added and visible in list. Default placeholder image shown. | Pass |
| UT26 | Verify that adding a product assigns the correct product ID prefix based on its category | 1. Select category from dropdown.<br>2. Enter valid product info.<br>3. Tap Save. | Category: Ingredients<br>Name: Purple Taro Balls<br>Qty: 25<br>Min Stock Qty: 20 | Success alert shown. In product list, productId starts with ING. New product visible under “Ingredients” tab. | Pass |

Table 7. 8: Unit Test Case of View Product Detail.

| Test Case Title | View Product Detail | | Test Module | Inventory Tracking | |
|---|---|---|---|---|---|
| Test Case ID | TC07 | | | | |
| Pre-conditions | From Products list, open any product’s Detail screen. | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT-27 | View product details | 1. Open Products list. | - | Detail screen opens | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | 2. Tap a product card. | | showing Product name, ID, Category, Quantity, Min stock qty, Stock status and Product image or placeholder. | |

Table 7. 9: Unit Test Case of Edit Product Info.

| Test Case Title | Edit Product Info | | Test Module | Inventory Tracking | |
|---|---|---|---|---|---|
| Test Case ID | TC08 | | | | |
| Pre-conditions | From Products list, open any product's Detail screen. | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT28 | Edit product name and category | 1. Tap Edit icon.<br>2. Change product name and category.<br>3. Tap Save.<br>4. Confirm changes. | Old: 700ml Shaker, Cat: Equipment.<br>New: 500ml Shaker, Cat: Packaging. | Confirmation popup shown. Success alert displayed ("Product updated"). Screen updates to show new name and category. | Pass |
| UT29 | Cancel edit | 1. Tap Edit icon. | Change name to "Test | Discard confirmation | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | 2. Change fields.<br>3. Tap Cancel.<br>4. Confirm discard. | Item". | popup appears. On confirm, changes not saved. Detail screen remains with original values. | |

Table 7. 10: Unit Test Case of Stock Update Using Product In.

| Test Case Title | Stock Update using Product In | | Test Module | Inventory Tracking | |
|---|---|---|---|---|---|
| Test Case ID | TC09 | | | | |
| Pre-conditions | From Products list, open any product's Detail screen. | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT30 | Add stock to product | 1. Tap IN option.<br>2. Enter quantity to add.<br>3. Tap Add. | Current qty=16.<br>Add=5. | Success alert displayed: "Quantity updated". Product quantity updates to 21. | Pass |
| UT31 | Add stock that changes status from Low Stock to In Stock | 1. Tap IN option.<br>2. Enter quantity that raises stock above min. | Current qty = 10<br>Min stock = 20<br>Add = 15 | Success alert displayed. Quantity increases to 25. Status changes from | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | 3. Tap Add. | | Low Stock (red dot) to In Stock (no red dot). | |
| UT32 | Add stock that keeps status as Low Stock | 1. Tap IN option.<br>2. Enter small quantity that keeps stock below min.<br>3. Tap Add. | Current qty = 5<br>Min stock = 20<br>Add = 10 | Success alert displayed. Quantity increases to 15. Status remains Low Stock, red dot stays. | Pass |
| UT33 | Cancel adding stock | 1. Tap IN option.<br>2. Enter quantity.<br>3. Tap Cancel. | Any value | Stock unchanged. Screen remains with original quantity. No alert shown. | Pass |

Table 7. 11: Unit Test Case of Stock Update using Product Out.

| Test Case Title | Stock Update using Product Out | | Test Module | Inventory Tracking | |
|---|---|---|---|---|---|
| Test Case ID | TC10 | | | | |
| Pre-conditions | From Products list, open any product's Detail screen. | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT34 | Reduce stock | 1. Tap OUT option.<br>2. Enter quantity to | Current qty=16.<br>Subtract=5 | Confirmation popup shown. Success alert | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | subtract.<br>3. Tap Subtract. | | displayed. Product quantity decreases to 11. | |
| UT35 | Prevent negative stock | 1. Tap OUT option.<br>2. Enter qty larger than current stock.<br>3. Tap Subtract. | Current qty=5.<br>Subtract=10. | Alert popup shown: "Quantity cannot be negative." Quantity unchanged. | Pass |
| UT36 | Add stock that changes status from In Stock to Low Stock | 1. Tap OUT option.<br>2. Enter quantity that lowers stock below min.<br>3. Tap Subtract. | Current qty = 25<br>Min stock = 20<br>Subtract = 10 | Success alert displayed. Quantity updates to 15. Status indicator changes to Low Stock | Pass |
| UT37 | Reduce stock to 0 (Out of Stock) | 1. Tap OUT option.<br>2. Enter full qty value.<br>3. Tap Subtract. | Current qty = 8<br>Subtract = 8 | Success alert displayed. Quantity updates to 0. Status label changes to Out of Stock. | Pass |
| UT38 | Cancel reduce stock action | 1. Tap OUT option.<br>2. Enter any quantity.<br>3. Tap Cancel. | Qty input = 5 | Action cancelled. No alert shown. Quantity remains unchanged. | Pass |

Table 7. 12: Unit Test Case of Delete Product.

| Test Case Title | Delete Product | | Test Module | Inventory Tracking | |
|---|---|---|---|---|---|
| Test Case ID | TC11 | | | | |
| Pre-conditions | From Products list, open any product's Detail screen. | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT39 | Delete product from inventory | 1. On Detail screen, tap Delete option.<br>2. Confirm delete action. | Product: Bamboo Toothpick. | Delete confirmation popup and success alert "Product deleted" shown. Deleted product no longer visible in list. | Pass |

Table 7. 13: Unit Test Case of Camera Permission and Preview.

| Test Case Title | Camera Permission and Preview | | Test Module | Inventory Tracking | |
|---|---|---|---|---|---|
| Test Case ID | TC12 | | | | |
| Pre-conditions | User is logged in and navigates to Scan screen. | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |

| | | | | | |
|---|---|---|---|---|---|
| UT40 | Request camera permission | 1. Open Scan with no prior permission<br>2. Tap Allow Camera | - | Modal “Camera Access Needed” appears; after allowing, modal closes and camera preview with scan frame is shown. | Pass |
| UT41 | Show camera preview | 1. Open Scan screen with permission granted. | - | Fullscreen camera view renders with green-cornered square scan frame. | Pass |

Table 7. 14: Unit Test Case of Scan the QR code on inventory item.

| Test Case Title | Scan the QR code on inventory item | | Test Module | Inventory Tracking | |
|---|---|---|---|---|---|
| Test Case ID | TC13 | | | | |
| Pre-conditions | User is logged in and navigates to Scan screen. Camera permission granted. | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT42 | Scan valid product QR | 1. Point camera at valid QR. | {"productId":"ING123456", "name":"Longan"} | Longan product card appears with correct | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | 2. Wait for scan. | | ID, name, qty, status. | |
| UT43 | Scan unknown product, which is not in database | 1. Scan QR whose productId isn't stored | {"productId":"ING999999", "name":"Unknown"} | Alert "The product is not found. Please add it first." No card is added; | Pass |
| UT44 | Scan invalid QR (non-JSON) | 1. Point camera at invalid QR. | 1234567890 | Alert pops up "Scanned Barcode: 1234567890". User remains on Scan; no crash. | Pass |
| UT45 | Rescan after a scan | 1. Successfully scan a product.<br>2. Tap Tap to Scan Again. | Any valid QR | Camera preview reopens. | Pass |
| UT46 | Scan QR code to view scanned product card | 1. Land on Scanned Product Detail after valid scan. | {"productId":"ING123456", "name":"Longan"} | Card displays product image, ID, name, qty, and status. | Pass |
| UT47 | Scan same product QR twice | 1. Scan product A.<br>2. Scan product A again using 'Scan' icon button in scanned product | {"productId":"ING123456", "name":"Longan"}<br>{"productId":"ING123456", "name":"Longan"} | Only one card for Longan is shown. Duplicate scans of same item do not add | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | detail. | | multiple cards. | |
| UT48 | Adjust quantity using +/– | 1. On Scanned Product Detail, tap + twice then – once. | Default 1 → + → + → – | Quantity field updates to 2. | Pass |
| UT49 | Adjust quantity going below 1 | 1. On Scanned Product Detail, tap – repeatedly. | Default = 1 → – → – | Quantity cannot drop below 1. Field stays at 1. | Pass |
| UT50 | Adjust every item quantity | 1. Scan two products.<br>2. Tap + once for item A.<br>3. Tap + twice for item B. | Item A: 1 → 2<br>Item B: 1 → 3 | Each card updates independently. Item A shows qty 2. Item B shows qty 3. | |
| UT51 | Update a single scanned item with Product In (All) | 1. Scan only one item.<br>2. Set quantity to 3.<br>3. Tap Product In (All). | Add = 3 | Success alert: “Product added successfully”. The scanned list clears and app returns to Scan screen. | Pass |
| UT52 | Check the reset quantity after update | 1. Scan item A. | Qty set = 5 | After update, when item A is scanned | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | 2. Set quantity = 5.<br>3. Tap Product In (All).<br>4. Scan item A again. | | again, its quantity resets to default 1 in the new card. | |
| UT53 | Update multiple items with Product In (All) | 1. Scan item A and B.<br>2. Set A=2, B=5.<br>3. Tap Product In (All). | A add 2; B add 5 | Success alert shown, list clears and app navigates back to Scan. | Pass |
| UT54 | Update a single scanned item with Product Out (All) | 1. Scan item A.<br>2. Set quantity to 3.<br>3. Tap Product Out (All). | Current 5, out 3 | Success alert shown, list clears and app returns to Scan. | Pass |
| UT55 | Update multiple items with Product Out (All) | 1. Scan item with stock 10.<br>2. Set Out=4.<br>3. Tap Product Out (All). | Current 10, Out 4 | Success alert shown, list clears and app returns to Scan. No negative UI states. | Pass |
| UT56 | Scan one item and try to subtract more than current stock | 1. Scan one item.<br>2. Set Out = 5 while current stock = 3.<br>3. Tap Product Out (All). | Current 3, Out 5 | Alert displayed: “Cannot remove more than current stock for <product name>.” | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | | | No changes made and return to scan screen. | |
| UT57 | Scan multiple items and one item has subtraction larger than stock | 1. Scan item A<br>2. Scan item B<br>3. Set Out A = 5, Out B = 4.<br>4. Tap Product Out (All). | A: Current = 3, Out = 5<br>B: Current = 10, Out = 4 | Alert displayed: “Cannot remove more than current stock for <product A>.” Valid item B still processes and updates. | Pass |
| UT58 | Scan another QR from detail screen | 1. On Scanned Product Detail, tap the QR icon. | - | App navigates back to Scan and the camera preview is active for the next scan. | Pass |

#### 7.2.2.3 Load Planning Module

Table 7. 15: Unit Test Case of Navigate from Load Plan home.

| Test Case Title | Navigate from Load Plan home | Test Module | Load Planning |
|---|---|---|---|
| Test Case ID | TC14 | | |

| Pre-conditions | The user is logged in and on the Load Plan tab which shows three buttons labelled Plan Load, Set Common Size, and History. | | | | |
|---|---|---|---|---|---|
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT59 | Navigate to the Plan Load screen | 1. Tap the Plan Load button | - | The Plan Load screen appears with the Container section, the Cargo section, and the Confirm Selection button | Pass |
| UT60 | Navigate to the Set Common Size screen | 1. Tap the Set Common Size button | - | The Set Common Size screen appears with the Add New Container form and the Add New Cargo form | Pass |
| UT61 | Navigate to the History screen | 1. Tap the History button | - | The Load Plan History list appears with the search box and load plan history. | Pass |

Table 7. 16: Unit Test Case of Container and cargo selection.

| Test Case Title | Container and cargo selection | | Test Module | Load Planning | |
|---|---|---|---|---|---|
| Test Case ID | TC15 | | | | |
| Pre-conditions | The user is on the Plan Load screen. | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT62 | Show available container and cargo lists | 1. Open the Plan Load screen.<br>2. Tap '+' button of the container and cargo. | - | Displays a list of containers and cargo items. | Pass |
| UT63 | Selects one container | 1. Tap a container card | Container 1 | The selected container card is highlighted. | Pass |
| UT64 | Selects two container | 1. Tap one container card.<br>2. Tap another container card. | Container 1, Container 2 | Only the last selected container remains highlighted. | Pass |
| UT65 | Add one cargo item to the selection | 1. Tap a cargo card. | Cargo 2 | The cargo appears in the Selected Cargo panel with a default quantity of one. | Pass |

| | | | | | |
|---|---|---|---|---|---|
| UT66 | Add multiple cargo items to the selection | 1. Tap a cargo card.<br>2. Tap one more cargo card. | Cargo 2, Cargo 4 | Both cargo items appear in the Selected Cargo panel with default quantity of one. | Pass |
| UT67 | Adjusts the quantity of a selected cargo | 1. In the Selected Cargo panel, tap the '+' button twice.<br>2. Tap the '-' button once. | Default quantity = 1 | The quantity increases to 2 and cannot drop below 1. | Pass |
| UT68 | Add cargo that exceeds container weight capacity | 1. Select a container with maximum weight capacity of 1500 kg.<br>2. Add multiple cargoes with total weight of 1600 kg.<br>3. Tap Confirm Selection. | Container 1 (1500 kg), Cargo A + Cargo B = 1600 kg | An alert 'Total cargo weight exceeds container weight. Cannot add more' appears, and no further additions can be made. | Pass |

Table 7. 17: Unit Test Case of Confirmation Details and Actual Ratio Diagram.

| Test Case Title | Confirmation Details and Actual Ratio Diagram | Test Module | Load Planning |
|---|---|---|---|

| Test Case ID | TC16 | | | | |
|---|---|---|---|---|---|
| Pre-conditions | The user has selected at least one container and one cargo, then tapped “Confirm Selection.” | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT69 | Confirmation details show selected container and cargo information | 1. Select one container and one cargo.<br>2. Tap Confirm Selection. | Container 1: 420×200×220 cm, 1500 kg.<br>Cargo 1: 100×100×140 cm, 10 kg, Qty: 1 | Confirmation screen displays container name, dimensions, max weight, and cargo list with dimensions, weight, and quantity. | Pass |
| UT70 | Actual ratio diagram displays cargo placement proportionally | 1. Add multiple cargo items to a container.<br>2. Tap Confirm Selection. | Container 1. Cargo 2 quantity one. Cargo 4 quantity two. | Diagram renders blocks representing each cargo with distinct labels and sizes, fitting proportionally inside container area. | Pass |
| UT71 | Space utilization percentage is calculated | 1. Add cargos that partially fill container. | Container capacity = 8.4 m². Cargos total area = 3.0 m². | Space utilization shows “Used Space: 3.00 m² | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | correctly | 2. Tap Confirm Selection. | | (35.7%)" and "Free Space: 5.40 m² (64.3%)". | |
| UT72 | Handles exceeding cargo that does not fit visually | 1. Add a cargo larger than the container dimensions.<br>2. Tap Confirm Selection. | Container:420×200×220 cm<br>Cargo: 500×250×300 cm. | Cargo not added to diagram and appears under 'Cargo(es) not fit into container' section | Pass |

Table 7. 18: Unit Test Case of Generate and save load plan.

| Test Case Title | Generate and save load plan | | Test Module | Load Planning | |
|---|---|---|---|---|---|
| Test Case ID | TC17 | | | | |
| Pre-conditions | The user is on the Confirmation Details screen after selecting a container and one or more cargo items. The diagram and space summary are visible. | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT73 | Saves a valid load plan | 1. Add container and cargo items.<br>2. Update the quantity of cargo.<br>3. Tap Save Load | Container 1: 420×200×220 cm, 1500 kg. Cargo 2 qty 1 (10 kg), Cargo 4 qty 2 (200 kg each) | A success alert appears: "Load plan saved successfully." | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | Plan. | | | |
| UT74 | Saved plan is visible in History after success | 1. Save a valid plan.<br>2. Navigate to History. | Recently saved plan | A new card appears with an auto-incremented ID, container info, used and free space, date, and cargo summary. The status is Pending until arranged. | Pass |

Table 7. 19: Unit Test Case of View load plan history.

| Test Case Title | View load plan history | | Test Module | Load Planning | |
|---|---|---|---|---|---|
| Test Case ID | TC18 | | | | |
| Pre-conditions | At least one load plan may exist. The user is on the History screen. | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT75 | The list shows existing load plans with key fields | 1. Open the History screen. | Plans exist | The screen renders load plan cards with details. | Pass |
| UT76 | The screen shows an empty state when there are no plans | 1. Open History with no records. | No plans | Empty state appears that says “No load plans found” with no list items. | Pass |
| UT77 | Scroll to view more plans | 1. Open History with many | >10 plans | Additional cards appear smoothly | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | records.<br>2. Scroll down the list. | | as the user scrolls, no layout jumps or overlaps. | |
| UT78 | Search by an exact plan ID | 1. Type an existing ID in the search box. | ID: 4 | Only the card for ID: 4 remains in the list. | Pass |
| UT79 | Search by a non-existent ID | 1. Enter an ID that does not exist. | ID: 999 | No results found | Pass |
| UT80 | Tapping a card opens the plan detail | 1. Tap a plan card. | Any listed plan | The app navigates to Load Plan Detail for that ID. | Pass |
| UT81 | Opens a plan by scanning a valid QR code | 1. Tap the scan icon.<br>2. Scan a valid load plan QR. | Valid QR encodes plan ID 1 | The camera view closes and Load Plan Detail (ID: 1) opens. | Pass |
| UT82 | Handles an invalid QR code | 1. Tap the scan icon.<br>2. Scan a random QR not linked to any plan. | Random text | An alert appears ‘Not found’ | Pass |

Table 7. 20: Unit Test Case of View load plan detail.

| Test Case Title | View load plan detail | Test Module | Load Planning |
|---|---|---|---|
| Test Case ID | TC19 | | |

| Pre-conditions | The user has opened a load plan from the History screen or scanned a valid QR code. The Load Plan Detail screen is displayed. | | | | |
|---|---|---|---|---|---|
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT83 | Delete a plan | 1. Tap the Delete icon.<br>2. Confirm deletion in the popup. | Plan ID: 5 | A success alert appears and navigated back to the History screen where Plan ID 5 is no longer listed. | Pass |

Table 7. 21: Unit Test Case of Arrange cargo checklist.

| Test Case Title | Arrange cargo checklist | | Test Module | Load Planning | |
|---|---|---|---|---|---|
| Test Case ID | TC20 | | | | |
| Pre-conditions | The user opened a pending load plan from History or scanned a valid QR code and navigated to the Arrange Cargo screen. | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT84 | Marks all cargo as arranged and saves | 1. Tap each cargo row to mark as arranged.<br>2. Tap Save (Settled). | Cargo A qty 2, Cargo B qty 3 | The Save (Settled) button becomes enabled only when all cargo are marked. After tapping | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | | | Save, the plan status changes to Finished in History. | |
| UT85 | Prevents saving if some cargoes are not arranged | 1. Mark only one cargo row as arranged.<br>2. Try tapping Save | Cargo A arranged, Cargo B not arranged | The Save (Settled) button remains disabled. | Pass |
| UT86 | Save action updates the plan status | 1. Arrange all cargo.<br>2. Tap Save (Settled).<br>3. Return to History. | Plan ID: 11 | The plan card in History now shows status as Finished. | Pass |
| UT87 | Cancel arrangement | 1. Return to previous page before marking cargo. | Any pending plan | The status remains Pending. | Pass |

Table 7. 22: Unit Test Case of Set common size for containers and cargo.

| Test Case Title | Set common size for containers and cargo | | Test Module | Load Planning | |
|---|---|---|---|---|---|
| Test Case ID | TC21 | | | | |
| Pre-conditions | The user is on the Set Common Size screen. | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |

| | | | | | |
|---|---|---|---|---|---|
| UT88 | Add a new container with valid inputs | 1. Enter container name.<br>2. Enter length, width, and height in centimetres.<br>3. Enter maximum weight.<br>4. Tap Add Container. | Name: Container A. Size: 420 × 200 × 220 cm. Weight: 1500 kg. | A success alert appears and container is added to the list. | Pass |
| UT89 | Add a new cargo with valid inputs | 1. Enter cargo name.<br>2. Enter length, width, and height in centimetres.<br>3. Enter weight.<br>4. Tap Add Cargo. | Name: Cargo X. Size: 100 × 100 × 100 cm. Weight: 50 kg. | A success alert appears and cargo is added to the list. | Pass |

Table 7. 23: Unit Test Case of Export a plan to PDF.

| Test Case Title | Export a plan to PDF | | Test Module | Load Planning | |
|---|---|---|---|---|---|
| Test Case ID | TC22 | | | | |
| Pre-conditions | The user is on the Load Plan Detail screen and the plan has been saved previously. | | | | |
| Test Case ID | Test Case Description | Execution Steps | Test Data | Expected Result | Status |
| UT90 | Exports a plan to PDF | 1. Tap the Export to PDF button on the detail | Plan ID: 7 | A share or download option appears with the | Pass |

| | | | | | |
|---|---|---|---|---|---|
| | | screen. | | generated file. | |
| UT91 | Exported PDF contains all required details | 1. Tap the Export to PDF button.<br>2. Open the downloaded file. | Plan ID: 12 with saved diagram and cargo list. | The PDF includes the load plan diagram, QR code, container info, cargo list, used/free/ total space, and arrangement order. | Pass |

### 7.3 Integration Test

Integration testing is conducted to verify that the different modules of the system interact and work together as expected. It ensures that data flows correctly between modules, and that combined functionalities achieve the intended outcomes. For this project, integration tests cover end-to-end flows across the User Management, Inventory Tracking, and Load Planning modules, validating that the entire system operates seamlessly when modules are integrated.

Table 7. 24: Test Case of Integration Test.

| Test Case ID | Test Case Description | Execution Steps | Expected Result | Status |
|---|---|---|---|---|
| IT01 | Register a new user, sign in, scan multiple valid product | 1. Tap the + button to open Add New User modal. | · New user is created and can log in successfully. | Pass |

| | | | | |
|---|---|---|---|---|
| | QRs to perform Product Out, create a load plan, save it, see it in History, export to PDF, scan the PDF QR and start arranging | 2. Enter Name, Email, and Password.<br>3. Tap Add and wait for the success alert.<br>4. Navigate to Personal and tap Logout<br>5. On Login screen, enter the newly created Email and Password and tap Login.<br>6. After landing on Home, go to Personal tab.<br>7. Verify the displayed Name and ID match the created user.<br>8. Open Scan tab and scan QR of product A and B, set quantity.<br>9. Tap 'Product Out' button.<br>10. Open Products and verify both items' quantities decreased. | · Product stock decreases correctly after Product Out.<br>· Load Plan is generated with selected container and cargo, and diagram is shown.<br>· User can rearrange cargo and save the plan.<br>· History shows the saved plan with Pending status.<br>· Exported PDF contains load plan details with scannable QR.<br>· Scanning the QR reopens the same plan.<br>· Arranging all cargo updates the plan status to Finished. | |

| | | | | |
|---|---|---|---|---|
| | | 11. Go to Load Plan Tab and click Plan Load.<br>12. Select Container 1 and add Cargo A with quantity 5.<br>13. Tap Confirm Selection.<br>14. Drag and drop to change the cargo placement.<br>15. Save Load Plan.<br>16. Open History and locate the new plan card with Pending status.<br>17. Open the plan and tap Export to PDF.<br>18. From History, tap the scan icon and scan the QR in the PDF.<br>19. In the plan detail, open Arrange Cargo.<br>20. Tick all cargo items.<br>21. Tap Save (Settled). | | |

## 7.4 User Acceptance Test (UAT)

User Acceptance Testing (UAT) was conducted to ensure that the developed mobile application for inventory tracking and cargo load planning meets the specified functional requirements and delivers smooth user experience for its intended end users, such as storekeepers and logistics staff in small and medium-sized enterprises (SMEs).

A total of three testers from different backgrounds were selected to participate in the UAT sessions to simulate real-world users. These testers represented typical roles such as inventory staff and logistics workers who would normally handle daily stock operations and vehicle load planning. The testing sessions were conducted face-to-face, which allowed for direct observation of the testers' interactions with the application and provided immediate opportunities to collect feedback.

Each tester was provided with an android smartphone with the application. They were guided to perform all prepared test cases step by step, based on the detailed instructions outlined in the UAT form. The test cases covered the full workflow of the system, including registering and logging into an account, adding, editing, and deleting inventory items, updating stock quantities through manual entry or barcode scanning, generating cargo load plans automatically or via manual rearrangement, checking cargo arrangement through the checklist, exporting load plan reports in PDF format, and retrieving load plan history using QR code scanning.

The results of the UAT sessions were recorded, and the detailed results from each tester are attached in Appendix C. A results summary collected from all three testers is tabulated as follows.

Table 7. 25: UAT Result Summary.

| Tester | No. of Test Case Executed | Pass | Fail | Overall Result |
|---|---|---|---|---|
| Tester 1 | 15 | 15 | 0 | Pass |
| Tester 2 | 15 | 15 | 0 | Pass |
| Tester 3 | 15 | 15 | 0 | Pass |
| Total | 45 | 45 | 0 | All Passed |

Although all test cases were successfully passed, testers provided several valuable comments during the UAT sessions. These comments highlight areas where the system can be further improved to enhance usability, efficiency, and flexibility. The comments were analysed as follows.

Table 7. 26: Analysis of UAT Feedback.

| Test Case ID | Test Case Title | Tester Comment | Analysis |
|---|---|---|---|
| UAT01 | User Registration | Prefer to have different role like admin or staff to differentiate. | The system only provides a single user type because the project scope focused on core inventory and load planning functions. Adding role-based access control like staff and admin in the future would enhance security and assign responsibilities more clearly. For example, only Admin can delete items, Staff can update stock. |
| UAT03 | Update Stock Quantity | Can have push alert when the stock low. | Low stock is currently shown only with a red indicator. Implement a push notification or popup alert would ensure that stock replenishment is not missed, improving operational efficiency. |
| | | No history record to know who, when, how many stock updated before. | Now, stock updates are reflected but without an audit trail. Adding a stock update history log in the future. |
| UAT05 | View Inventory | Filter function is enough for this app, | Current filtering is limited to categories because only basic product |

| | | | |
|---|---|---|---|
| | List | but prefer more methods to search like by supplier, by expiry date in the future. | attributes are stored in the database. Supplier and expiry date were not required in the initial scope. In future versions, the product attributes can be extended to support advanced filtering options like supplier, expiry date, and storage location, allowing the system to be tailored more closely to the needs of different companies with larger inventories. |
| UAT07 | Edit Item | Product name and min stock qty should be edited as well. | Editing currently supports only certain attributes. However, in future versions, the editing function can be extended to include minimum stock quantity and other additional fields, allowing the system to be tailored to the specific needs of each company. |
| UAT09 | Generate Load Plan | The load plan diagram looks a bit crowded, suggest have zoom in or out. | The load plan diagram was designed to fit within a single screen to ensure compatibility and ease of use on Android phones. However, this approach limits visibility when dealing with complex load plans containing many cargoes. Zoom or rotate options can be implemented in future versions to enhance clarity and usability for more detailed arrangements. |
| UAT11 | Rearrange Cargo Manually | Drag and drop function works, but currently cargoes can overlap when | Overlap validation was not implemented to keep the drag and drop function simple. Preventing overlap requires boundary checks and |

| | | | |
|---|---|---|---|
| | | rearranged manually. | grid alignment logic, which can be added later to ensure realistic arrangements. |
| UAT13 | View Load Plan History | Prefer to have edit function so can rearrange the cargo placement, no need create a new one if using same cargoes. | Load plans are currently stored as final records for reference only. Adding an edit feature in the future versions would save time by letting users adjust existing plans without the need to recreate them from scratch. |
| | | The saved load plan cannot be edited. | |

Table 7. 27: User Acceptance Testing Form (UAT).

| **User Acceptance Testing Form (UAT)** | | | | | |
|---|---|---|---|---|---|
| **Name** | | | | | |
| **Role / Position** | | | | | |
| **Date of Testing** | | | | | |
| **Testing Start Time** | | **Testing End Time** | | | |
| **Test Case ID** | **Test Case Title** | **Test Steps** | **Expected Results** | **Status** | **Comments** |
| UAT01 | Register a new user | 1. Tap + in the User List Screen.<br>2. Enter Name, Email, Password.<br>3. Tap Add. | Success alert 'User added!' shown. New created user appears in the user list with correct name and email. | | |
| UAT02 | Login an account | 1. User enters the assigned email 'abc@gmail.com' and password '123456' in the login page.<br>2. User clicks on the login button. | Users will be navigated to Home Screen with welcome message. | | |
| UAT03 | Update an item's | 1. User clicks 'Product' tab. | A successful alert will be | | |

| | | | | | |
|---|---|---|---|---|---|
| | stock quantity | 2. Use ‘Search’ function in the inventory list to find the product to be adjusted.<br>3. Click the product card to opens the product's description screen.<br>4. Tap the ‘IN’ button, enter the quantity and confirms. | displayed. The product's inventory quantity and status will be updated in real time on the product’s description screen. | | |
| UAT04 | Update 3 items’ stock quantity | 1. User clicks the “Scan” tab.<br>2. The user scans the first item’s barcode and adjusts the quantity.<br>3. User clicks the ‘Scan’ icon at the right bottom side in the scan details screen and repeats step 2 until there are no more items to scan.<br>4. User clicks the ‘Product In’ | A successful alert will be displayed. The stock quantity of scanned items is updated accordingly. | | |

| | | | | | |
|---|---|---|---|---|---|
| | | button. | | | |
| UAT05 | View inventory list | 1. Navigate to the Products tab from the bottom menu.<br>2. View the complete list of inventory items displayed on the screen.<br>3. Use the category tabs or filter option to display products by category.<br>4. Select an inventory card to open and view its detailed information. | Product Detail screen showing all details of that product. | | |
| UAT06 | Add a new item | 1. The user clicks the "+" button on the inventory list screen.<br>2. The user fills in the required fields such as Product image, Product name, Current stock quantity, Category, and | A success message is displayed: "Item added successfully" and the new item appears in the inventory list | | |

| | | | | | |
|---|---|---|---|---|---|
| | | Minimum stock quantity<br>3. The user clicks the “Save” button. | | | |
| UAT07 | Edit an item | 1. User selects the item to be edited from the inventory.<br>2. Clicks the ‘three dots’ icon at the top right corner and selects the ‘edit’ button.<br>3. Change product name and category.<br>4. Tap Save to confirm changes. | Success alert displayed ‘Product updated’. Screen updates to show new name and category. | | |
| UAT08 | Delete an inventory item | 1. The user selects the item to be deleted from the inventory.<br>2. The user clicks the ‘three dots’ icon at the top right corner and selects the ‘trash bin’ icon button.<br>3. The staff confirms the deletion | A success message is displayed ‘Item deleted successfully’ and returns back to the inventory list screen. | | |

| | | | | | |
|---|---|---|---|---|---|
| | | by clicking ‘Delete’. | | | |
| UAT09 | Generate Load Plan using custom container size and cargo dimension | 1. The user taps the Load Plan tab on the navigation bar.<br>2. The user selects the Set Common Size option.<br>3. The user enters the container dimensions and maximum load capacity.<br>4. The user enters the cargo dimensions.<br>5. The user returns to the Load Plan tab and selects Plan Load.<br>6. The user chooses the container and cargo added, then adjusts the quantities.<br>7. The user taps Confirm Selection.<br>8. The user reviews the generated | New container and cargo are added into the system. A load plan is generated and saved. | | |

| | | | | | |
|---|---|---|---|---|---|
| | | load plan and taps Save. | | | |
| UAT10 | Generate Load Plan using previously defined container size and cargo dimensions | 1. The user taps the Load Plan tab on the navigation bar.<br>2. The user selects the Plan Load option.<br>3. The user chooses a previously saved container from the list.<br>4. The user chooses one or more previously saved cargo items.<br>5. The user adjusts the cargo quantities as needed.<br>6. The user taps Confirm Selection.<br>7. The user reviews the generated load plan and taps Save. | A load plan with diagram and space usage is generated and saved. | | |
| UAT11 | Rearranging cargo item manually in a load plan | 1. The user opens the Load Plan tab.<br>2. The user selects a container and | Cargo items can be rearranged manually in the diagram. The updated | | |

| | | | | | |
|---|---|---|---|---|---|
| | | cargo, then generates a load plan.<br>3. On the Confirmation Details screen, the user drags and drops cargo items to rearrange their positions.<br>4. The user taps the Save button. | arrangement is saved successfully. | | |
| UAT12 | Generate PDF report | 1. The user opens the History tab.<br>2. The user selects a saved load plan from the list.<br>3. On the Load Plan Detail screen, the user taps the Export to PDF button.<br>4. The system generates the PDF and displays the option to view or download it. | A PDF report is generated successfully. The PDF can be opened or downloaded. | | |
| UAT13 | View Load Plan History | 1. The user navigates to the History tab. | The corresponding Load Plan Detail screen is | | |

| | | | | | |
|---|---|---|---|---|---|
| | | 2. The user searches by Load Plan ID or uses the filter function to filter by status.<br>5. The user taps the Load Plan card from the list. | displayed with complete information. | | |
| UAT14 | View Load Plan Details by Scanning QR in PDF | 1. The user navigates to the History tab.<br>2. On the Load Plan History screen, the user taps the Scan button.<br>3. The user scans the QR code on the printed PDF. | The QR code on the printed PDF is scanned successfully. The system opens the corresponding Load Plan Detail screen. | | |
| UAT15 | Arrange Cargo Checklist | 1. The user opens the History tab.<br>2. The user selects a load plan with status Pending.<br>3. On the Load Plan Detail screen, the user taps the Arrange Cargo option. | Cargo items can be marked as arranged using the checklist. The system updates the load plan status to Finished after saving. | | |

| | | | | | |
|---|---|---|---|---|---|
| | | 4. The user marks each cargo item in the checklist as arranged.<br>3. The user taps the Save (Settled) button. | | | |

## 7.5 System Usability Test (SUS)

In addition to User Acceptance Testing (UAT), a System Usability Test (SUS) was conducted to evaluate the overall usability and user-friendliness of the developed mobile application. While UAT focuses on validating whether the system meets functional requirements, SUS is designed to measure users' subjective perception of the system in terms of ease of use, efficiency, and learnability.

The SUS questionnaire consisted of ten standardized statements rated on a five-point Likert scale, ranging from Strongly Disagree (1) to Strongly Agree (5). To conduct the SUS, each tester was given the questionnaire immediately after completing the UAT session to ensure their feedback was based on fresh experience. The same three testers who participated in UAT also took part in the SUS evaluation, providing a consistent perspective across both tests.

The testers completed the SUS individually and without external influence, so their answers reflected their personal opinions. Once collected, the responses were tabulated and scored using the standard SUS calculation method. Each odd-numbered question was scored as the user's response minus 1, while each even-numbered question was scored as 5 minus the user's response. The values were then summed up for each tester and multiplied by 2.5 to convert the raw score into a usability score out of 100. Finally, the three testers' scores were averaged to determine the overall SUS score of the system.

Table 7. 28: Template of SUS form.

| Participant No: | | | | | |
|---|---|---|---|---|---|
| Name | | | | | |
| Question | Strongly Disagree | | | | Strongly Agree |
| | 1 | 2 | 3 | 4 | 5 |
| 1. I think that I would like to use this inventory and load planning app frequently. | | | | | |
| 2. I found the app unnecessarily complex | | | | | |

| | | | | | |
|---|---|---|---|---|---|
| when managing inventory or planning loads. | | | | | |
| 3. I thought the app was easy to use. | | | | | |
| 4. I think that I would need the support of a technical person to be able to use this app. | | | | | |
| 5. I found the barcode scanning, inventory, and load planning functions in this app were well integrated. | | | | | |
| 6. I experienced inconsistencies in the app (e.g., navigation, layout, or functions) that made it harder to use. | | | | | |
| 7. I believe new users can learn to use this app without much difficulty. | | | | | |
| 8. I found the app cumbersome to use when performing tasks like scanning or arranging cargo. | | | | | |
| 9. I felt very confident using the app to manage inventory and generate load plans. | | | | | |
| 10. I needed to learn a lot of things before I could start using this app effectively. | | | | | |

1. What do you like most about the app?
2. What did you like the least about the app?
3. Did you face any bugs, errors, or unexpected behavior while using the system? If yes, please describe.
4. Do you have any suggestions for improving the system?

The results of the System Usability Test (SUS) show that the three testers gave overall usability scores of 87.5%, 87.5%, and 80.0%, with an average SUS score of 85.0%. According to the standard SUS benchmark scale, a score above 68 is considered above average, while scores above 80 are regarded as excellent. Therefore, the obtained score of 85 places the developed system in the “Excellent Usability” category (Grade A).

This result indicates that testers found the system easy to use, efficient, and user-friendly, with minimal difficulties in performing tasks such as scanning items, updating stock, and generating load plans. While the UAT comments highlighted some areas for improvement, the high SUS score confirms that the system is already highly usable and well-accepted by its intended users.

Table 7. 29: Summary of SUS Survey Results.

| **Tester** | **Usability score for each question** | | | | | | | | | | **Total** | **Percentage (%)** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | | |
| 1 | 3 | 4 | 4 | 4 | 4 | 3 | 3 | 4 | 3 | 3 | 35 | 87.5 |
| 2 | 4 | 4 | 4 | 4 | 3 | 3 | 3 | 3 | 4 | 3 | 35 | 87.5 |
| 3 | 3 | 3 | 4 | 4 | 5 | 3 | 3 | 3 | 3 | 3 | 32 | 80 |
| **Average SUS Score** | | | | | | | | | | | | **85** |
| **Grade** | | | | | | | | | | | | **A** |

Apart from the standard System Usability Scale (SUS) questionnaire, several open-ended questions were also included to give testers the opportunity to share their personal opinions about the system. The summary of their responses is shown below, highlighting what they liked most, what they liked least, any issues they faced, and their suggestions for improvement.

The testers highlighted several key strengths of the system, particularly the features that made their daily work faster and easier. Their comments are summarized in the table below.

Table 7. 30: Summary of Testers' Feedback on the Most Liked Features.

| What testers liked most about the app: |
| --- |
| The scan function is useful and quick to tracking item. My company currently use Excel to record stock quantity, which is time-consuming and sometime have typo mistakes. So having the scanning barcode allows the stock to be updated instantly in the system. |
| The automatic arranges cargo feature. It is better than draw one by one cargo using Excel and fit it to the container, as my company currently does, take around 1 hours for 1 plan. With this app, it auto generates the arrangement, I just check only, maybe 5 minutes can finish already. |
| Overall the system is good, easy to use and quite straightforward. |

Across all three testers, no significant weaknesses or bugs were reported. Two testers explicitly stated that they had no dislikes, while one mentioned that the system overall worked smoothly without errors or crashes. This feedback indicates that the system was generally stable and well-received, with no major usability concerns raised.

Although the testers were satisfied with the system overall, the testers also provided constructive suggestions for future enhancements. Their feedback is summarized in the table below.

Table 7. 31: Summary of Testers' Suggestions for System Improvement.

| Suggestions for improvement: |
| --- |
| I think it's better if got history record for stock updates, so can know who update the stock, when update, and how many change each time. In business this is important, because sometimes stock got wrong number, then very hard to find why. If got history, manager can trace back to see who update wrong and correct it faster. |
| The load plan diagram all shows in one screen, so a bit crowded and hard to see clearly. If can add zoom in or out, then easier or maybe got computer version to do it also better. |
| No |

# CHAPTER 8

# CONCLUSION

## 8.1 Conclusion

This project focuses on developing a mobile application for inventory tracking with barcode scanning and cargo load planning optimization. The main motivation is to overcome the inefficiencies of traditional manual or Excel-based methods, which are often time-consuming and prone to human error. By combining inventory management and load planning into a single application, the system provides a faster, more accurate, and more user-friendly solution for small and medium-sized enterprises (SMEs), particularly for storekeepers and logistics staff.

In the initial phase, the project focuses on identifying the problem, defining objectives, and gathering requirements. A literature review is conducted to study traditional inventory methods, manual load planning techniques, and existing applications, while interviews and observations are used to validate the practical needs of storekeepers and logistics staff. Requirements are then modelled using use cases, interface flows, and initial prototypes.

The middle phase concentrates on designing and implementing the system. The architecture is structured into clear modules, supported by Firebase Firestore for real-time data management. Features such as user management, inventory tracking with barcode scanning, cargo load planning with both automated algorithms and manual drag-and-drop, as well as report generation with PDF and QR code support, are developed.

Finally, in the last phase, the system is tested and validated. Unit testing and integration testing confirm the correctness of individual modules and their interactions. User Acceptance Testing (UAT) is carried out with three testers, all of whom pass successfully. The System Usability Test (SUS) further validates the user-friendliness of the application, producing an excellent average usability score of 85%. Feedback collected highlights strengths such as the barcode scanning and automatic cargo arrangement

features, while also pointing out potential improvements including stock history tracking and enhanced load plan visualization.

In summary, the project successfully achieves its objectives and delivers a functional mobile application that integrates inventory management with cargo load planning in a practical and efficient way.

## 8.2 Objective Achievements

The project set out three main objectives as outlined in Chapter 1, all of which were successfully achieved during the development and testing process.

1. To conduct a thorough study of algorithms for generating optimal cargo load plans for vehicles.
2. To develop a functional mobile app for inventory tracking with integrated cargo load planning and optimization features for vehicle space utilization.
3. To evaluate the developed mobile app with Unit Test, System Usability Scale (SUS) and User Acceptance Testing (UAT).

The first objective is achieved through an in-depth literature review of existing load planning algorithms, including the Biased Random-Key Genetic Algorithm (BRKGA) and the Binary Tree Bin Packing algorithm. After comparison, the Binary Tree Bin Packing algorithm is selected and implemented in the system because it provides faster and more consistent results with lower computational complexity, making it more suitable for SMEs and mobile environments than BRKGA.

The second objective is fully accomplished with the successful development of a mobile application using React Native and Firebase Firestore. The application integrates multiple modules, such as user management, inventory tracking with barcode scanning, stock quantity updates, load plan generation using the selected algorithm, manual cargo rearrangement through drag-and-drop, and load plan reporting through PDF export and QR code scanning. Together, these features form a practical and functional solution that

enables SMEs to manage inventory effectively while optimizing vehicle space utilization.

Lastly, the third objective is also achieved. Unit testing is carried out to validate the correctness of individual modules, while integration testing confirms smooth interaction between system components. User Acceptance Testing (UAT) is conducted with three testers, covering 15 test cases, all of which pass successfully. In addition, a System Usability Test (SUS) is performed, which results in an excellent usability score of 85% (Grade A). This outcome demonstrates that the developed system is not only functional but also user-friendly, efficient, and well-accepted by its intended end users.

## 8.3 Limitations and Recommendations of Future Work

Although the developed mobile application successfully meets its stated objectives, there are still several limitations in its current version. These limitations arise mainly from the simplified assumptions made during development, the scope constraints defined at the start of the project, and the need to balance usability with technical complexity. To address these issues, several recommendations have been identified for future improvements, which are outlined together with the limitations in the following table.

Table 8. 1: Limitation and Recommendations.

| | Limitations | Recommendations |
|---|---|---|
| 1 | **No user role differentiation**<br>At present, all users share the same access level and functions. This may not reflect real business environments, where admin, managers, and staff need different levels of access and control. | Implement a role-based access control system where admin can manage users and perform critical actions, while staff are limited to day-to-day operations. |
| 2 | **Cargo assumptions and constraints**<br>The system assumes all cargo items are rectangular or square, | Enhance the algorithm to support irregular and cylindrical cargo shapes and integrate rules for weight distribution, fragility, and |

| | | |
|---|---|---|
| | and it does not account for irregular shapes, cylindrical items, weight distribution, fragility, or stacking limitations. This may lead to wasted space or unsafe arrangements in real-world use. | stacking restrictions. This will make the system more realistic and suitable for complex logistics environments. |
| 3 | **No history record for stock updates**<br>The system does not keep track of who updated the stock, when it was updated, and how much was changed. This reduces transparency and makes it difficult to trace errors. | Add a stock update history log that records each update with details such as user, date or time, and change in quantity. This feature help track errors and provide a more realistic record for business operations. |
| 4 | **Manual drag-and-drop cargo arrangement allows overlap**<br>Currently, users can place cargo items on top of each other, which is not realistic and may result in confusion. | Implement boundary checks and snap-to-grid alignment to prevent overlapping cargo. This will ensure more accurate arrangements and better reflect physical loading constraints. |
| 5 | **Load plan diagram can be crowded on mobile screens**<br>When multiple cargoes are displayed in one container, the diagram becomes difficult to read on smaller devices. | Add zoom and rotate functions to improve visibility. Additionally, consider developing a desktop or web-based version of the system for complex load plans where more screen space is required. |

‘

## REFERENCES


Asia-Pacific Economic Cooperation. (2017). *Current Situation of Heavy Vehicle Overloading in Malaysia*. [online] Available at: https://mddb.apec.org/Documents/2017/TPTWG/WKSP1/17_%20tptwg_wksp1_018.pdf.

Barratt, M., Kull, T.J. and Sodero, A.C. (2018). Inventory record inaccuracy dynamics and the role of employees within multi-channel distribution center inventory systems. *Journal of Operations Management*, 63(1), pp.6–24. doi: https://doi.org/10.1016/j.jom.2018.09.003.

BYJUS. (n.d.). *Cube and Cuboid Shape (Definition, Formulas & Properties)*. [online] Available at: https://byjus.com/maths/cuboid-and-cube/.

Chan, C.W., Sathiapriya, A.R. and Razali, N.F. (2023). Inventory Management Systems (IMS). *Journal of Applied Technology and Innovation*, [online] 7(3), pp.2600–7304. doi: http://dx.doi.org/10.1088/1742-6596/1573/1/012038.

Chopra, C. (2021). *Why Manual Inventory Tracking Must Be Replaced with Automated Inventory Tracking?* [online] Infizo. Available at: https://www.infizo.com/stock/blog/manual-vs-automated-inventory-tracking.

Douglas, C. (2025). *An Ultimate Guide to Throwaway Prototyping - Visily*. [online] www.visily.ai. Available at: https://www.visily.ai/blog/throwaway-prototyping/.

Gaur, P. (2023). *The Importance of Load Planning: Tips for Efficient and Safe Shipment*. [online] Cargoflip.com. Available at: https://www.cargoflip.com/post/load-planning#google_vignette.

GeeksforGeeks. (2025). *Software Engineering | Incremental process model - GeeksforGeeks*. [online] Available at: https://www.geeksforgeeks.org/software-engineering-incremental-process-model/.

Gonçalves, J.F. and Resende, M.G.C. (2013). A biased random key genetic algorithm for 2D and 3D bin packing problems. *International Journal of Production Economics*, [online] 145(2), pp.500–510. doi: https://doi.org/10.1016/j.ijpe.2013.04.019.

Gordon, J. (2011). *Binary Tree Bin Packing Algorithm*. [online] Codeincomplete.com. Available at: https://codeincomplete.com/articles/bin-packing/.

Kuhn, J. (2021). The Financial Impact of Manual Inventory Record Errors. *International Journal of Business and Social Science*, [online] 12(10), pp.4–5. doi: https://doi.org/10.30845/ijbss.v12n10p2.

Londe, M.A., Pessoa, L.S., Andrade, C.E. and Resende, M.G.C. (2024). Biased random-key genetic algorithms: A review. *European Journal of Operational Research*, 321(1), pp.1–22. doi: https://doi.org/10.1016/j.ejor.2024.03.030.

Madamidola, O.A., Daramola, O., Akintola, K. and Adeboje, O. (2024). A Review of Existing Inventory Management Systems. *International Journal of Research in Engineering and Science*, [online] 12(9), pp.40–50. Available at: https://www.researchgate.net/publication/383947700_A_Review_of_Existing_Inventory_Management_Systems.

Ministry of Transport Malaysia (2021). *Land Transportation Acts*. [online] Mot.gov.my. Available at: https://www.mot.gov.my/en/land/acts-and-regulations/land-transportation.

Perera, M., Hettiarachchi, M., Pabasara, T., Kulasekara, V. and Parindya, J. (2022). Study on Throwaway Prototyping Model over PcD.UcT Model. *SW Modelling CI Wk2*, [online] p.2. doi: https://doi.org/10.13140/RG.2.2.27238.50243.

Saraiva, R.D., Nepomuceno, N. and Pinheiro, P.R. (2015). A layer-building algorithm for the three-dimensional multiple bin packing problem: a case study in an automotive company. *IFAC-PapersOnLine*, 48(3), pp.490–495. doi: https://doi.org/10.1016/j.ifacol.2015.06.129.

Setrag Shahikian (2024). *Finale Inventory*. [online] Finale Inventory. Available at: https://www.finaleinventory.com/inventory-management/manual-vs-automated-inventory-management-key-distinctions-explained-ecommerce.

# APPENDICES

## Appendix A: Interview Questions.

**Project Title: Developing an App for Streamlined Inventory Tracking with Barcode Scanning and Load Planning Optimization**

**Interview Questions**

**Section 1: General Information**

1. Can you introduce yourself and your role in your company?
2. How many years of experience do you have in your career?
3. What type of inventory does your company manage?
4. Does your company currently use any inventory management system? If yes, which one?

**Section 2: Inventory Tracking**

5. How does your company currently track inventory (Barcode scanning system/ Spreadsheets (e.g., Excel, Google Sheets) / Inventory Management Software)?
6. How often do you conduct inventory audits?
7. How do you ensure inventory data accuracy and prevent discrepancies?

**Section 3: Barcode Scanning Integration**

8. Do you currently use barcode scanning in your inventory tracking? If yes, how effective is it?
   *If the answer to Question 8 is yes, proceed to Questions 9-12; otherwise, skip to Question 13*
9. What type of barcode scanners (Handheld / Mobile App-based / Fixed-mount) or technology do you use?
10. What challenges do you face when using barcode scanning for inventory tracking?
11. Are there any specific software limitations you face with your current barcode scanning system?
12. What features or improvements would make barcode scanning more efficient for your company?

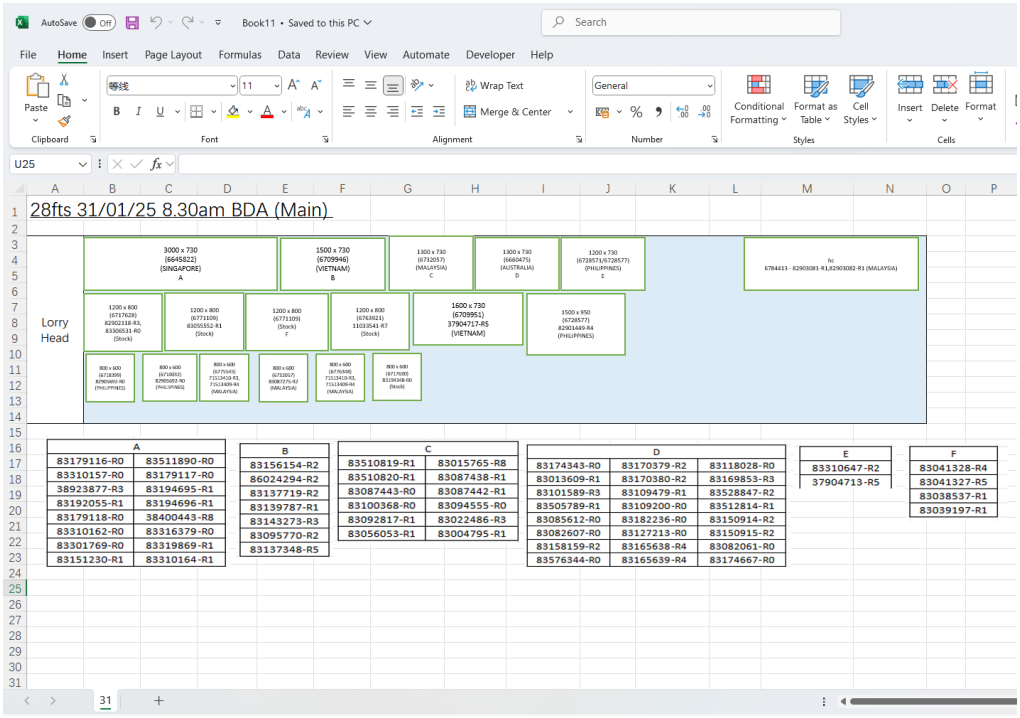
**Section 4: Load Planning Optimization**

13. How does your company currently handle the placement of cargo into containers or vehicles?
14. What are the key factors you consider when planning load optimization (e.g., weight distribution, space utilization, delivery schedules)?
15. What challenges do you face in load planning?
16. How much time do you typically spend on load planning?
17. Would your organization be interested in a system that optimizes load planning automatically?

**Section 5: User Experience & App Development**

18. What features would you expect in an inventory tracking and load planning app?
19. Do you require integration with existing systems (e.g., ERP, warehouse management software)?
20. What platforms (mobile, tablet, desktop) would you prefer for using this application?
21. Are there any additional suggestions for improving an inventory tracking app? Please specify below.
22. Would you be open to testing a prototype and providing feedback during the development process?

# Appendix B: Manual Sketch of Cargo Layout in Excel.



Appendix C-1: User Acceptance Testing Result of Tester 1.

| **User Acceptance Testing Form (UAT)** | | | | | |
|---|---|---|---|---|---|
| **Name** | Yuki | | | | |
| **Role / Position** | Warehouse Manager of 4 Beans Cafe | | | | |
| **Date of Testing** | 9/9/2025 | | | | |
| **Testing Start Time** | 9.33am | **Testing End Time** | 10.57am | | |
| **Test Case ID** | **Test Case Title** | **Test Steps** | **Expected Results** | **Status** | **Comments** |
| UAT01 | Register a new user | 1. Tap + in the User List Screen.<br>2. Enter Name, Email, Password.<br>3. Tap Add. | Success alert 'User added!' shown. New created user appears in the user list with correct name and email. | Pass | |
| UAT02 | Login an account | 1. User enters the assigned email 'abc@gmail.com' and password '123456' in the login page.<br>2. User clicks on the login button. | Users will be navigated to Home Screen with welcome message. | Pass | |
| UAT03 | Update an item's stock quantity | 1. User clicks 'Product' tab.<br>2. Use 'Search' function in the | A successful alert will be displayed. The product's | Pass | No history record to know |

| | | | | | |
|---|---|---|---|---|---|
| | | inventory list to find the product to be adjusted.<br>3. Click the product card to opens the product's description screen.<br>4. Tap the 'IN' button, enter the quantity and confirms. | inventory quantity and status will be updated in real time on the product's description screen. | | who, when, how many stock updated before. |
| UAT04 | Update 3 items' stock quantity | 1. User clicks the "Scan" tab.<br>2. The user scans the first item's barcode and adjusts the quantity.<br>3. User clicks the 'Scan' icon at the right bottom side in the scan details screen and repeats step 2 until there are no more items to scan.<br>4. User clicks the 'Product In' button. | A successful alert will be displayed. The stock quantity of scanned items is updated accordingly. | Pass | |
| UAT05 | View inventory list | 1. Navigate to the Products tab | Product Detail screen | Pass | Filter function is |

| | | | | | |
|---|---|---|---|---|---|
| | | from the bottom menu.<br>2. View the complete list of inventory items displayed on the screen.<br>3. Use the category tabs or filter option to display products by category.<br>4. Select an inventory card to open and view its detailed information. | showing all details of that product. | | enough for this app, but prefer more methods to search like by supplier, by expiry date in the future. |
| UAT06 | Add a new item | 1. The user clicks the "+" button on the inventory list screen.<br>2. The user fills in the required fields such as Product image, Product name, Current stock quantity, Category, and Minimum stock quantity.<br>3. The user clicks the 'Save' | A success message is displayed: "Item added successfully" and the new item appears in the inventory list | Pass | |

| | | | | | |
|---|---|---|---|---|---|
| | | button. | | | |
| UAT07 | Edit an item | 1. User selects the item to be edited from the inventory.<br>2. Clicks the ‘three dots’ icon at the top right corner and selects the ‘edit’ button.<br>3. Change product name and category.<br>4. Tap Save to confirm changes. | Success alert displayed ‘Product updated’. Screen updates to show new name and category. | Pass | Min stock qty should be edited as well. |
| UAT08 | Delete an inventory item | 1. The user selects the item to be deleted from the inventory.<br>2. The user clicks the ‘three dots’ icon at the top right corner and selects the ‘trash bin’ icon button.<br>3. The staff confirms the deletion by clicking ‘Delete’. | A success message is displayed ‘Item deleted successfully’ and returns back to the inventory list screen. | Pass | |
| UAT09 | Generate Load Plan | 1. The user taps the Load Plan tab | New container and cargo | Pass | |

| | | | | | |
|---|---|---|---|---|---|
| | using custom container size and cargo dimension | on the navigation bar.<br>2. The user selects the Set Common Size option.<br>3. The user enters the container dimensions and maximum load capacity.<br>4. The user enters the cargo dimensions.<br>5. The user returns to the Load Plan tab and selects Plan Load.<br>6. The user chooses the container and cargo added, then adjusts the quantities.<br>7. The user taps Confirm Selection.<br>The user reviews the generated load plan and taps Save. | are added into the system. A load plan is generated and saved. | | |
| UAT10 | Generate Load Plan | 1. The user taps the Load Plan tab | A load plan with diagram | Pass | |

| | | | | | |
|---|---|---|---|---|---|
| | using previously defined container size and cargo dimensions | on the navigation bar.<br>2. The user selects the Plan Load option.<br>3. The user chooses a previously saved container from the list.<br>4. The user chooses one or more previously saved cargo items.<br>5. The user adjusts the cargo quantities as needed.<br>6. The user taps Confirm Selection.<br>7. The user reviews the generated load plan and taps Save. | and space usage is generated and saved. | | |
| UAT11 | Rearranging cargo item manually in a load plan | 1. The user opens the Load Plan tab.<br>2. The user selects a container and cargo, then generates a load plan. | Cargo items can be rearranged manually in the diagram. The updated arrangement is saved successfully. | Pass | |

| | | | | | |
|---|---|---|---|---|---|
| | | 3. On the Confirmation Details screen, the user drags and drops cargo items to rearrange their positions.<br>4. The user taps the Save button. | | | |
| UAT12 | Generate PDF report | 1. The user opens the History tab.<br>2. The user selects a saved load plan from the list.<br>3. On the Load Plan Detail screen, the user taps the Export to PDF button.<br>4. The system generates the PDF and displays the option to view or download it. | A PDF report is generated successfully. The PDF can be opened or downloaded. | Pass | |
| UAT13 | View Load Plan History | 1. The user navigates to the History tab.<br>2. The user searches by Load Plan ID or uses the filter function to | The corresponding Load Plan Detail screen is displayed with complete information. | Pass | The saved load plan cannot be edited. |

| | | | | | |
|---|---|---|---|---|---|
| | | filter by status.<br>3. The user taps the Load Plan card from the list. | | | |
| UAT14 | View Load Plan Details by Scanning QR in PDF | 1. The user navigates to the History tab.<br>2. On the Load Plan History screen, the user taps the Scan button.<br>3. The user scans the QR code on the printed PDF. | The QR code on the printed PDF is scanned successfully. The system opens the corresponding Load Plan Detail screen. | Pass | |
| UAT15 | Arrange Cargo Checklist | 1. The user opens the History tab.<br>2. The user selects a load plan with status Pending.<br>3. On the Load Plan Detail screen, the user taps the Arrange Cargo option.<br>4. The user marks each cargo item in the checklist as arranged. | Cargo items can be marked as arranged using the checklist. The system updates the load plan status to Finished after saving. | Pass | |

| | | | | | |
|---|---|---|---|---|---|
| | | 5. The user taps the Save (Settled) button. | | | |

Appendix C-2: User Acceptance Testing Result of Tester 2.

| **User Acceptance Testing Form (UAT)** | | | | | |
|---|---|---|---|---|---|
| **Name** | Ms Jesther | | | | |
| **Role / Position** | Logistics Assistant Manager, Ametal Tech Sdn. Bhd. | | | | |
| **Date of Testing** | 11/9/2025 | | | | |
| **Testing Start Time** | 10.14am | **Testing End Time** | 12.07pm | | |
| **Test Case ID** | **Test Case Title** | **Test Steps** | **Expected Results** | **Status** | **Comments** |
| UAT01 | Register a new user | 1. Tap + in the User List Screen.<br>2. Enter Name, Email, Password.<br>3. Tap Add. | Success alert 'User added!' shown. New created user appears in the user list with correct name and email. | Pass | |
| UAT02 | Login an account | 1. User enters the assigned email 'abc@gmail.com' and password '123456' in the login page.<br>2. User clicks on the login button. | Users will be navigated to Home Screen with welcome message. | Pass | |
| UAT03 | Update an item's stock quantity | 1. User clicks 'Product' tab.<br>2. Use 'Search' function in the | A successful alert will be displayed. The product's | Pass | |

| | | | | | |
|---|---|---|---|---|---|
| | | inventory list to find the product to be adjusted.<br>3. Click the product card to opens the product's description screen.<br>4. Tap the 'IN' button, enter the quantity and confirms. | inventory quantity and status will be updated in real time on the product's description screen. | | |
| UAT04 | Update 3 items' stock quantity | 1. User clicks the "Scan" tab.<br>2. The user scans the first item's barcode and adjusts the quantity.<br>3. User clicks the 'Scan' icon at the right bottom side in the scan details screen and repeats step 2 until there are no more items to scan.<br>4. User clicks the 'Product In' button. | A successful alert will be displayed. The stock quantity of scanned items is updated accordingly. | Pass | |
| UAT05 | View inventory list | 1. Navigate to the Products tab | Product Detail screen | Pass | |

| | | | | | |
|---|---|---|---|---|---|
| | | from the bottom menu.<br>2. View the complete list of inventory items displayed on the screen.<br>3. Use the category tabs or filter option to display products by category.<br>4. Select an inventory card to open and view its detailed information. | showing all details of that product. | | |
| UAT06 | Add a new item | 1. The user clicks the "+" button on the inventory list screen.<br>2. The user fills in the required fields such as Product image, Product name, Current stock quantity, Category, and Minimum stock quantity.<br>3. The user clicks the 'Save' | A success message is displayed: "Item added successfully" and the new item appears in the inventory list | Pass | |

| | | | | | |
|---|---|---|---|---|---|
| | | button. | | | |
| UAT07 | Edit an item | 1. User selects the item to be edited from the inventory.<br>2. Clicks the ‘three dots’ icon at the top right corner and selects the ‘edit’ button.<br>3. Change product name and category.<br>4. Tap Save to confirm changes. | Success alert displayed ‘Product updated’. Screen updates to show new name and category. | Pass | |
| UAT08 | Delete an inventory item | 1. The user selects the item to be deleted from the inventory.<br>2. The user clicks the ‘three dots’ icon at the top right corner and selects the ‘trash bin’ icon button.<br>3. The staff confirms the deletion by clicking ‘Delete’. | A success message is displayed ‘Item deleted successfully’ and returns back to the inventory list screen. | Pass | |
| UAT09 | Generate Load Plan | 1. The user taps the Load Plan tab | New containers and | Pass | The load plan |

| | | | | | |
|---|---|---|---|---|---|
| | using custom container size and cargo dimension | on the navigation bar.<br>2. The user selects the Set Common Size option.<br>3. The user enters the container dimensions and maximum load capacity.<br>4. The user enters the cargo dimensions.<br>5. The user returns to the Load Plan tab and selects Plan Load.<br>6. The user chooses the container and cargo added, then adjusts the quantities.<br>7. The user taps Confirm Selection.<br>The user reviews the generated load plan and taps Save. | cargo are added into the system. A load plan is generated and saved. | | diagram looks a bit crowded, suggest have zoom in or out. |
| UAT10 | Generate Load Plan | 1. The user taps the Load Plan tab | A load plan with diagram | Pass | |

| | | | | | |
|---|---|---|---|---|---|
| | using previously defined container size and cargo dimensions | on the navigation bar.<br>2. The user selects the Plan Load option.<br>3. The user chooses a previously saved container from the list.<br>4. The user chooses one or more previously saved cargo items.<br>5. The user adjusts the cargo quantities as needed.<br>6. The user taps Confirm Selection.<br>7. The user reviews the generated load plan and taps Save. | and space usage is generated and saved. | | |
| UAT11 | Rearranging cargo item manually in a load plan | 1. The user opens the Load Plan tab.<br>2. The user selects a container and cargo, then generates a load plan. | Cargo items can be rearranged manually in the diagram. The updated arrangement is saved successfully. | Pass | Drag and drop function works, but currently cargoes can overlap when |

| | | | | | |
|---|---|---|---|---|---|
| | | 3. On the Confirmation Details screen, the user drags and drops cargo items to rearrange their positions.<br>4. The user taps the Save button. | | | rearranged manually. |
| UAT12 | Generate PDF report | 1. The user opens the History tab.<br>2. The user selects a saved load plan from the list.<br>3. On the Load Plan Detail screen, the user taps the Export to PDF button.<br>4. The system generates the PDF and displays the option to view or download it. | A PDF report is generated successfully. The PDF can be opened or downloaded. | Pass | |
| UAT13 | View Load Plan History | 1. The user navigates to the History tab.<br>2. The user searches by Load Plan ID or uses the filter function to | The corresponding Load Plan Detail screen is displayed with complete information. | | Prefer to have edit function so can rearrange the cargo placement, |

| | | | | | |
|---|---|---|---|---|---|
| | | filter by status.<br>3. The user taps the Load Plan card from the list. | | | no need create a new one if using same cargoes. |
| UAT14 | View Load Plan Details by Scanning QR in PDF | 1. The user navigates to the History tab.<br>2. On the Load Plan History screen, the user taps the Scan button.<br>3. The user scans the QR code on the printed PDF. | The QR code on the printed PDF is scanned successfully. The system opens the corresponding Load Plan Detail screen. | Pass | |
| UAT15 | Arrange Cargo Checklist | 1. The user opens the History tab.<br>2. The user selects a load plan with status Pending.<br>3. On the Load Plan Detail screen, the user taps the Arrange Cargo option.<br>4. The user marks each cargo item in the checklist as arranged. | Cargo items can be marked as arranged using the checklist. The system updates the load plan status to Finished after saving. | Pass | |

| | | | | | |
|---|---|---|---|---|---|
| | | 5. The user taps the Save (Settled) button. | | | |

Appendix C-3: User Acceptance Testing Result of Tester 3.

| **User Acceptance Testing Form (UAT)** | | | | | |
|---|---|---|---|---|---|
| **Name** | Mr Luo | | | | |
| **Role / Position** | Worker of TaiYi Machinery Equipment Co., Ltd. | | | | |
| **Date of Testing** | 15/9/2025 | | | | |
| **Testing Start Time** | 3.49pm | **Testing End Time** | 4.22pm | | |
| **Test Case ID** | **Test Case Title** | **Test Steps** | **Expected Results** | **Status** | **Comments** |
| UAT01 | Register a new user | 1. Tap + in the User List Screen.<br>2. Enter Name, Email, Password.<br>3. Tap Add. | Success alert 'User added!' shown. New created user appears in the user list with correct name and email. | Pass | Prefer to have different role like admin or staff to differentiate. |
| UAT02 | Login an account | 1. User enters the assigned email 'abc@gmail.com' and password '123456' in the login page.<br>2. User clicks on the login button. | Users will be navigated to Home Screen with welcome message. | Pass | |
| UAT03 | Update an item's stock quantity | 1. User clicks 'Product' tab.<br>2. Use 'Search' function in the | A successful alert will be displayed. The product's | Pass | Can have push alert when the |

| | | | | | |
|---|---|---|---|---|---|
| | | inventory list to find the product to be adjusted.<br>3. Click the product card to opens the product's description screen.<br>4. Tap the 'IN' button, enter the quantity and confirms. | inventory quantity and status will be updated in real time on the product's description screen. | | stock low. |
| UAT04 | Update 3 items' stock quantity | 1. User clicks the "Scan" tab.<br>2. The user scans the first item's barcode and adjusts the quantity.<br>3. User clicks the 'Scan' icon at the right bottom side in the scan details screen and repeats step 2 until there are no more items to scan.<br>4. User clicks the 'Product In' button. | A successful alert will be displayed. The stock quantity of scanned items is updated accordingly. | Pass | |
| UAT05 | View inventory list | 1. Navigate to the Products tab | Product Detail screen | Pass | |

| | | | | | |
|---|---|---|---|---|---|
| | | from the bottom menu.<br>2. View the complete list of inventory items displayed on the screen.<br>3. Use the category tabs or filter option to display products by category.<br>4. Select an inventory card to open and view its detailed information. | showing all details of that product. | | |
| UAT06 | Add a new item | 1. The user clicks the "+" button on the inventory list screen.<br>2. The user fills in the required fields such as Product image, Product name, Current stock quantity, Category, and Minimum stock quantity.<br>3. The user clicks the 'Save' | A success message is displayed: "Item added successfully" and the new item appears in the inventory list | Pass | |

| | | | | | |
|---|---|---|---|---|---|
| | | button. | | | |
| UAT07 | Edit an item | 1. User selects the item to be edited from the inventory.<br>2. Clicks the ‘three dots’ icon at the top right corner and selects the ‘edit’ button.<br>3. Change product name and category.<br>4. Tap Save to confirm changes. | Success alert displayed ‘Product updated’. Screen updates to show new name and category. | Pass | |
| UAT08 | Delete an inventory item | 1. The user selects the item to be deleted from the inventory.<br>2. The user clicks the ‘three dots’ icon at the top right corner and selects the ‘trash bin’ icon button.<br>3. The staff confirms the deletion by clicking ‘Delete’. | A success message is displayed ‘Item deleted successfully’ and returns back to the inventory list screen. | Pass | |
| UAT09 | Generate Load Plan | 1. The user taps the Load Plan tab | New container and cargo | Pass | |

| | | | | | |
|---|---|---|---|---|---|
| | using custom container size and cargo dimension | on the navigation bar.<br>2. The user selects the Set Common Size option.<br>3. The user enters the container dimensions and maximum load capacity.<br>4. The user enters the cargo dimensions.<br>8. The user returns to the Load Plan tab and selects Plan Load.<br>9. The user chooses the container and cargo added, then adjusts the quantities.<br>10. The user taps Confirm Selection.<br>The user reviews the generated load plan and taps Save. | are added into the system. A load plan is generated and saved. | | |
| UAT10 | Generate Load Plan | 1. The user taps the Load Plan tab | A load plan with diagram | Pass | |

| | | | | | |
|---|---|---|---|---|---|
| | using previously defined container size and cargo dimensions | on the navigation bar.<br>2. The user selects the Plan Load option.<br>3. The user chooses a previously saved container from the list.<br>4. The user chooses one or more previously saved cargo items.<br>5. The user adjusts the cargo quantities as needed.<br>6. The user taps Confirm Selection.<br>7. The user reviews the generated load plan and taps Save. | and space usage is generated and saved. | | |
| UAT11 | Rearranging cargo item manually in a load plan | 1. The user opens the Load Plan tab.<br>2. The user selects a container and cargo, then generates a load plan. | Cargo items can be rearranged manually in the diagram. The updated arrangement is saved successfully. | Pass | |

| | | | | | |
|---|---|---|---|---|---|
| | | 3. On the Confirmation Details screen, the user drags and drops cargo items to rearrange their positions.<br>4. The user taps the Save button. | | | |
| UAT12 | Generate PDF report | 1. The user opens the History tab.<br>2. The user selects a saved load plan from the list.<br>3. On the Load Plan Detail screen, the user taps the Export to PDF button.<br>4. The system generates the PDF and displays the option to view or download it. | A PDF report is generated successfully. The PDF can be opened or downloaded. | Pass | |
| UAT13 | View Load Plan History | 1. The user navigates to the History tab.<br>2. The user searches by Load Plan ID or uses the filter function to | The corresponding Load Plan Detail screen is displayed with complete information. | | |

| | | | | | |
|---|---|---|---|---|---|
| | | filter by status.<br>3. The user taps the Load Plan card from the list. | | | |
| UAT14 | View Load Plan Details by Scanning QR in PDF | 1. The user navigates to the History tab.<br>2. On the Load Plan History screen, the user taps the Scan button.<br>3. The user scans the QR code on the printed PDF. | The QR code on the printed PDF is scanned successfully. The system opens the corresponding Load Plan Detail screen. | Pass | |
| UAT15 | Arrange Cargo Checklist | 1. The user opens the History tab.<br>2. The user selects a load plan with status Pending.<br>3. On the Load Plan Detail screen, the user taps the Arrange Cargo option.<br>4. The user marks each cargo item in the checklist as arranged. | Cargo items can be marked as arranged using the checklist. The system updates the load plan status to Finished after saving. | Pass | |

| | | | | | |
|---|---|---|---|---|---|
| | | 5. The user taps the Save (Settled) button. | | | |

Appendix D-1: SUS Test Result of Tester 1.

| Participant No: | 1 | | | | |
|---|---|---|---|---|---|
| Name: | Yuki | | | | |
| Question | Strongly Disagree | | | | Strongly Agree |
| | 1 | 2 | 3 | 4 | 5 |
| 1. I think that I would like to use this inventory and load planning app frequently. | | | | ✓ | |
| 2. I found the app unnecessarily complex when managing inventory or planning loads. | ✓ | | | | |
| 3. I thought the app was easy to use. | | | | | ✓ |
| 4. I think that I would need the support of a technical person to be able to use this app. | ✓ | | | | |
| 5. I found the barcode scanning, inventory, and load planning functions in this app were well integrated. | | | | | ✓ |
| 6. I experienced inconsistencies in the app (e.g., navigation, layout, or functions) that made it harder to use. | | ✓ | | | |
| 7. I believe new users can learn to use this app without much difficulty. | | | | ✓ | |
| 8. I found the app cumbersome to use when performing tasks like scanning or arranging cargo. | ✓ | | | | |
| 9. I felt very confident using the app to manage inventory and generate load plans. | | | | ✓ | |
| 10. I needed to learn a lot of things before I could start using this app | | ✓ | | | |

| | | | | | |
|---|---|---|---|---|---|
| effectively. | | | | | |

1. What do you like most about the app?

The scan function is useful and quick to tracking item. My company currently use Excel to record stock quantity, which is time-consuming and sometime have typo mistakes. So having the scanning barcode allows the stock to be updated instantly in the system.

2. What did you like the least about the app?

No

3. Did you face any bugs, errors, or unexpected behaviour while using the system? If yes, please describe.

No, overall worked smoothly.

4. Do you have any suggestions for improving the system?

I think it's better if got history record for stock updates, so can know who update the stock, when update, and how many change each time. In business this is important, because sometimes stock got wrong number, then very hard to find why. If got history, manager can trace back to see who update wrong and correct it faster.

Appendix D-2: SUS Test Result of Tester 2.

| Participant No: | 2 | | | | |
|---|---|---|---|---|---|
| Name | Ms Jesther | | | | |
| Question | Strongly Disagree | | | | Strongly Agree |
| | 1 | 2 | 3 | 4 | 5 |
| 1. I think that I would like to use this inventory and load planning app frequently. | | | | | ✓ |
| 2. I found the app unnecessarily complex when managing inventory or planning loads. | ✓ | | | | |
| 3. I thought the app was easy to use. | | | | | ✓ |
| 4. I think that I would need the support of a technical person to be able to use this app. | ✓ | | | | |
| 5. I found the barcode scanning, inventory, and load planning functions in this app were well integrated. | | | | ✓ | |
| 6. I experienced inconsistencies in the app (e.g., navigation, layout, or functions) that made it harder to use. | | ✓ | | | |
| 7. I believe new users can learn to use this app without much difficulty. | | | | ✓ | |
| 8. I found the app cumbersome to use when performing tasks like scanning or arranging cargo. | | ✓ | | | |
| 9. I felt very confident using the app to manage inventory and generate load plans. | | | | | ✓ |
| 10. I needed to learn a lot of things before I could start using this app | | ✓ | | | |

| | | | | | |
|---|---|---|---|---|---|
| effectively. | | | | | |

1. What do you like most about the app?

The automatic arranges cargo feature. It is better than draw one by one cargo using Excel and fit it to the container, as my company currently does, take around 1 hours for 1 plan. With this app, it auto generates the arrangement, I just check only, maybe 5 minutes can finish already.

2. What did you like the least about the app?

None

3. Did you face any bugs, errors, or unexpected behavior while using the system? If yes, please describe.

No

4. Do you have any suggestions for improving the system?

The load plan diagram all shows in one screen, so a bit crowded and hard to see clearly. If can add zoom in or out, then easier or maybe got computer version to do it also better.

Appendix D-3: SUS Test Result of Tester 3.

| Participant No: | 3 | | | | |
|---|---|---|---|---|---|
| Name | Mr Luo | | | | |
| Question | Strongly Disagree | | | | Strongly Agree |
| | 1 | 2 | 3 | 4 | 5 |
| 1. I think that I would like to use this inventory and load planning app frequently. | | | | ✓ | |
| 2. I found the app unnecessarily complex when managing inventory or planning loads. | | ✓ | | | |
| 3. I thought the app was easy to use. | | | | | ✓ |
| 4. I think that I would need the support of a technical person to be able to use this app. | ✓ | | | | |
| 5. I found the barcode scanning, inventory, and load planning functions in this app were well integrated. | | | | ✓ | |
| 6. I experienced inconsistencies in the app (e.g., navigation, layout, or functions) that made it harder to use. | | ✓ | | | |
| 7. I believe new users can learn to use this app without much difficulty. | | | | ✓ | |
| 8. I found the app cumbersome to use when performing tasks like scanning or arranging cargo. | | ✓ | | | |
| 9. I felt very confident using the app to manage inventory and generate load plans. | | | | ✓ | |
| 10. I needed to learn a lot of things before I could start using this app | | ✓ | | | |

| | | | | | |
|---|---|---|---|---|---|
| effectively. | | | | | |

1. What do you like most about the app?

Overall the system is good, easy to use and quite straightforward.

2. What did you like the least about the app?

No

3. Did you face any bugs, errors, or unexpected behaviour while using the system? If yes, please describe.

No

4. Do you have any suggestions for improving the system?

No